UCRL-MA-127443 Rev. 1

MeshTV Command Line Interface Manual

March 2002

Version 4.3

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

Table of Contents

MeshTV Command Line Interface Manual

1.0 Overview

MeshTV is an interactive program for visualizing and analyzing scientific data. MeshTV reads SILO[1] data files, which allows it to run on many hardware platforms. MeshTV handles 1D, 2D, and 3D data and provides a variety of operations on the data, enabling it to be both a visualization tool and a data analysis program. A full description of the SILO data format for MeshTV can be found in the MeshTV-SILO Manual[2], and instructions for writing SILO files can be found in the SILO User's Guide[3].

While MeshTV has a Command Line Interface (CLI), most people prefer to use MeshTV's Graphical User Interface (GUI), which is described in the MeshTV User's Manual[4]. If you are just starting MeshTV for the first time, you might want to use the GUI for awhile to get a feeling for how the program works. You might also want to read the MeshTV Getting Started Manual[5]. The MeshTV Getting Started Manual describes important MeshTV concepts and assumptions, and it also contains a tutorial for the GUI.

This manual details the various commands supported by MeshTV as they are accessed via the Command Line Interface. Users might want to use the CLI for production (batch) jobs, or to access commands which have yet to be implemented in the GUI.

---

1. A high-level, portable interface that was developed at Lawrence Livermore National Laboratory to address difficult database issues, such as different, incompatible file formats and libraries, most of which used non-standard features of the Cray compilers.
2. MeshTV-SILO Manual, To Be Released.
3. SILO User's Guide, UCRL-MA-118751
4. MeshTV User's Manual, UCRL-MA-132625
5. MeshTV Getting Started Manual, UCRL-MA-127442

2.0 Conventions

The following typographical conventions are used in the command summaries:

italic	An option or parameter to be replaced by the user with an appropriate value.
`typeface`	Command names and options. These are given to MeshTV exactly as shown.
[ ]	Arguments enclosed in square brackets are optional. The brackets are not part of the argument.
|	The “pipe” symbol indicates “or”, so if two arguments are separated by the pipe symbol, this means one or the other is appropriate.

Many of the commands use commas to separate options, as in:

```
bnd var=material, lc=red
```

These commas are optional.

Real numbers ending in zero can be entered with or without the decimal point. For example, 0 can be entered as `0`, `0.`, or `0.0`.

3.0 MeshTV Execute Line

MeshTV can be run in command line mode (without the GUI). When used in this way, the program name is **`meshtvx`** rather than **`meshtv`**. To run MeshTV in command line mode, type the following at a UNIX™ prompt:

`meshtvx` *[options]*

options are:

`-banner`	Assigns banners for PostScript™ output, but only when the `-nowin` and `-ps` options are used. The banner to print must immediately follow the `-banner` option, and it must contain no spaces, unless the string is in quotes. For example:
	`meshtvx -nowin -ps foo -banner Print_This`
	`meshtvx -nowin -ps bar -banner “Print This”`

`-geometry`	Set the size of the MeshTV visualization window. Specify the height, width, and the x,y position of the top left corner. For example, 500X500+300+0.

`-log` *`filename`*	Specify a filename to which the MeshTV log will be written. This defaults to "`%meshtv.log`".
`-nowin`	Do not open visualization windows. This is useful for the production (batch) mode of operation.
`-n`	End the MeshTV prompt with a newline character. This is useful when MeshTV is run from within a script.
`-noinfo`	Do not output MeshTV informational messages.
`-nowarn`	Do not output MeshTV warning messages.
`-ppm` *rootname*	Write all plots in ppm format to the file family, the first file of which will be named *rootname*0000.ppm.
`-ps` *rootname*	Write all plots in PostScript™ format to the file family, the first file of which will be named *rootname*0000.ps.
`-q`	Quiet mode. Do not output informational or warning messages. This is equivalent to using the `-noinfo` and `-nowarn` options.
`-res` *number*	The X and Y resolutions for saved and printed images. In order to specify different X and Y resolutions use the `-xres` and `-yres` options. Used only with the `-ps`, `-rgb`, `-rps`, or `-tif` options.
`-rgb` *rootname*	Write all plots in SGI rgb format to the file family, the first file of which will be named *rootname*0000.rgb.
`-rps` *rootname*	Write all plots in Raster PostScript™ format to the file family, the first file of which will be named *rootname*0000.ps.
`-s` *filename*	Specify a filename from which to read commands. *filename* is a file containing MeshTV commands.
`-tif` *rootname*	Write all plots in tiff format to the file family, the first file of which will be named *rootname*0000.tif.
`-xres` *number*	The X resolution for saved and printed images. Used only with the `-ps`, `-rgb`, `-rps`, or `-tif` options.
`-yres` *number*	The Y resolution for saved and printed images. Used only with the `-ps`, `-rgb`, `-rps`, or `-tif` options.
file	Open the SILO file named *file*. Multiple files can be specified if they are members of the same family. Files in the same family share the same base name followed by increasing numbers, as in file0000.silo, file0001.silo, and file1001.silo.

initfile	Open a MeshTV init file. The init file which is always opened if it exists in the user's home directory is .meshtvinit. Other init files can be created and specified on the command line. Multiple files can be specified. An init file must begin with a '#', and it contains valid MeshTV commands. A pre-existing init file at some locations is "cale", so issuing the following command, `meshtvx cale`, would give MeshTV a Cale feel. Cale is a physics simulation code under development by Lawrence Livermore National Laboratory.

Commands contained in a file can be input to MeshTV via the C Shell redirection facility.

`meshtvx` *[options]* **<** *command_file*

Commands generated from a program can be piped to MeshTV.

program **|** `meshtvx` *[options]*

4.0 MeshTV Operators

MeshTV's operators are accessed via the `defvar` command and the `pop` option, rather than via separate commands of their own. The following table lists MeshTV's operators and the corresponding arguments to the appropriate command.

Operation	Command	Argument
Arbitrary slice	`defvar`	`aslice`
Erase	`pop`	`clipp`
Index select	`defvar`	`insel`
Orthogonal slice	`defvar`	`oslice`
Reflect (the data)	`defvar`	`reflect`
Reflect (the picture)	`pop`	`reflectp`
Reflect (another method)	`pop`	`symmetry`
Segment	`defvar`	`segment`

5.0 MeshTV Production Mode of Operation

MeshTV can be run in a production (batch) environment where a script program, for example, supplies plot commands in a file from which MeshTV generates plots as output files of various formats. In this case, no visualization window needs to exist, so the `-nowin` option might be used. The `source` command invoked from within MeshTV, or

the redirection of a command file on the execute line, can be used to supply the plot commands. The `foutput` command or the `-ps` execute line option can be used to direct plots to a PostScript™ file. Some examples of production mode usage follow.

5.1 MeshTV Example: nowin, ps, redirection

```
meshtvx -nowin -ps prob prob.silo <prob.log >/dev/null
```

Commands in file prob.log and SILO file prob.silo are used to generate plots which are written to the PostScript™ file family which starts with prob0000.ps. Plots are not sent to a visualization window. All MeshTV messages are redirected to the “great bit bucket in the sky” (/dev/null) rather than being sent to the user.

5.2 MeshTV Example: nowin, -n, interactive

```
meshtvx -nowin -n
1-->
foutput rootname=prob2
2-->
open prob200000.silo
Current DB is: prob200000.silo
‘2D rectilinear file’
3-->
source prob2.log
4-->
open prob200100.silo
Current DB is: prob200100.silo
‘2D rectilinear file’
5-->
source prob2.log
6-->
quit
```

Commands in file prob2.log and data in SILO files prob200000.silo and prob200100.silo are used to generate plots which are written to the PostScript™ file family which starts with prob20000.ps. Plots are not sent to a visualization window. The newline character ends the MeshTV prompt.

5.3 MeshTV Example: nowin, -q, interactive

```
meshtvx -nowin -q
1-->open prob3.silo
2-->foutput rootname=prob3
3-->source prob3.log
4-->foutput rootname=prob4
5-->source prob4.log
```

```
6-->quit
```

Commands in file prob3.log and data in the SILO file prob3.silo are used to generate plots which are written to the PostScript™ file family which begins with prob30000.ps. Ditto for prob4. Plots are not sent to a visualization window. The prompt does not include the newline character. No warnings or informational messages are sent.

If invalid commands are sent to MeshTV, an error message will be returned to the standard error ("standard err"). This message will begin with the string "MeshTV Error:"

Whenever MeshTV is run, whether from the command line or via the GUI, it will output a log of the commands it received during the session. This log will be named "`%meshtv.log`" You can use this log to rerun MeshTV, either from within the code by issuing the `source` command, or by using the redirection method mentioned earlier. The only restriction is that you must rename the file, or MeshTV will attempt to read from and write to the same file, and that would cause problems.

6.0 Command Summaries

The command summaries which follow describe the usage of every command in MeshTV. Each command summary is broken into five sections — Synopsis, Arguments, Description, Examples, and See Also.

The Synopsis section contains a brief summary of the command along with its arguments. The Arguments section contains a description of each argument supported by the command. The Description section provides a more thorough explanation of what the command does. The Examples section demonstrates sample uses of the command. The See Also section refers the reader to any related commands or documents which might also be of interest.

Note that these commands can be issued via the GUI by using the Command Line window. If these commands are issued from the GUI, all text output will go to the Output window. If these commands are issued from the Command Line Interface version of MeshTV, output goes to the standard output ("standard out"), which is usually the shell window in which MeshTV was invoked.

If you issue these commands from the Command Line Interface window in the GUI, the GUI's other windows will not correctly reflect the changes. For example, if you type "animate fps=5.0", the animate window will not reflect that change. In general, if you use the GUI, you should issue commands via the GUI's windows whenever possible.

addframe — Add a frame to the list of frames in an animation.

Synopsis:

`addframe` *filename*

Arguments:

filename	A SILO filename, including either absolute or relative path specifications.

Description:

Add a frame into the list of frames in an animation. The new frame will be added at the end of the list of frames. It is not possible to add a frame in the middle of a sequence of frames. To do so, you must start a new animation.

Examples:

```
open abc1.silo
addframe /usr/people/sample/abc2.silo
addframe ../data/abc3.silo
addframe ~/data/abc4.silo
```

See Also:

`animate, newplot, play, powerwall, rplay, setframe, stop`

alias — Define or list command aliases.

Synopsis:

`alias` *[*`type=`*type]*
`alias` *name*
`alias` `command=yes`
`alias` *[*`command=yes|no`*] name "value"*

Arguments:

name	An alphanumeric string, starting with a letter.
`command`	This indicates whether or not the alias is a command. If so, the alias should only be expanded when it appears as a command and not when it appears as a variable name. For example, consider the following alias: `alias ucdmesh “mesh var=ucdmesh1; plot mesh”` In this case, `command` is equal to yes (the default), so ucdmesh will be treated as a variable when you type “`mesh var=ucdmesh`”. If “`command=no`” is added to the above alias, when you type “`mesh var=ucdmesh`”, the ucdmesh is expanded into the command listed above, and there would be an error.
`type`	The “category” of aliases to which this alias belongs. By default, aliases for which a type is unspecified are in the “user” category. Other categories include “bdiv”, “cale” and “system”. Users may also create their own category types. This argument is used to display aliases by category. For example, a user might want to see what all the “cale” aliases are.
"*value*"	A quoted string containing MeshTV commands.

Description:

With the `alias` command, you choose a “name” to represent a string of MeshTV commands or aliases.

The `alias` command works in much the same way as it does under the C-Shell in UNIX™, where aliases (names) represent other operations. This allows the user to effectively rename existing commands, and to define a single command that executes numerous other commands.

With no arguments, or with just the `type` option, `alias` lists various aliases along with their values. By default, the `type` is “user” if left unspecified, so just typing “`alias`” will list those aliases created by the user. This is equivalent to typing “`alias type=user`”. Similarly, other groups of aliases can be listed. For example, to see the

system aliases that MeshTV contains, type "`alias type=system`". Likewise, to see the cale aliases or the bdiv aliases, type "`alias type=cale`" or "`alias type=bdiv`". Users can also create personal types. For example, if I am working on a project called CoolThing, I can define the following alias: "`alias type=CoolThing cool ls`" This would create an alias named "cool" that lists my data file. I can now type "`alias type=CoolThing`" to see all the aliases associated with the CoolThing project.

With one argument, `alias` prints the value associated with the given *name* argument. For example, based on the above paragraph, typing "`alias cool`" would yield a description of the cool alias.

By default, an alias is treated as a command, so the alias name is only expanded when the name occurs as a command and not when it occurs as a variable. Adding "`command=no`" indicates that the alias is not a command, so it will be expanded whenever it occurs on the command line. Note that if the above cool alias had been "`alias type=CoolThing command=no cool ls`", typing "`alias cool`" would yield an error because cool would be expanded to ls before being passed to the alias command. Since ls isn't an alias, MeshTV would report an error.

Command-only aliases (the default, or those with `command=yes` in their definition) can expand positional variables. Two kinds of positional variables are allowed, named and numerical. Numerical variables start with 1, and represent the position of the argument. So $1 would be the first numbered argument to the command, and $2 would be the second numbered argument. The positions for the numbered variables do not count named variables. Look at the following example:

```
alias command=yes pli "iso var=$1 lt=$width"
pli width=2 d
```

In the above example, $width is a named variable, and $1 is a numbered variable. Even though the "d" is the second argument it fits the $1 variable since it is the first numbered argument. This format allows the user to also type "`pli d width=2`"

Examples:

```
alias command=yes clr "clear"# Define an alias
alias clr      # Print alias definition for clr
alias          # Print definition of type=user aliases
alias command=no# Include command=no alias definitions
alias command=yes# Print type=user command=yes definitions
alias command=yes type=cale
alias type=system# Print all system aliases
alias plt "plot"
alias command=no type=user ucd "ucdmesh"
alias type=mytype command=yes doit "ls"
alias foo "clr; plt"
alias cplt "clear; plot"
```

```
# The following example is read from an init file.
alias type=cale command=yes plv " \
     if (defined \"vec_ppp\")                        \
          (error \"The vector plot is already plotted.\") \
          (progn                                      \
               (delete_vec)                           \
               (alias command=no \"vec_ppp\" yes) \
               (set_vec_symm)                         \
               (vec var=$1)                           \
               (plot_vec))                            \
";
```

See Also:

defined, error, if, progn, unalias

animate — Set animation options.

Synopsis:

`animate` *option=value [, option=value]*

Arguments:

option	One of the names of an animation property. See table below.
value	A value associated with the matching *option*. See table below

Option	Value	Value Meaning	Default Value
`cache`	`on, off`	Cache polygons in the machine's RAM. This speeds up animations, but requires large amounts of memory.	`off`
`fps`	real number >= 0.0	The number of frames per second. Setting the rate higher than the machine can draw the images will cause frames to be skipped. 0 has a special meaning; it indicates that the animation should go as fast as the machine allows.	`0.0`

Description:

Set options for the animation. `cache` speeds up the animation significantly, but it can only be used if your machine has sufficient memory, probably a minimum of 40 MB, though the actual number might be more or less than 40 since this number depends on the size of the problem and the number of frames in the animation. `fps` provides control over the frames per second.

Examples:

```
animate fps=5.0, cache=on
```

See Also:

`addframe, newplot, play, powerwall, rplay, setframe, stop`

apropos — Print information about the keyword.

Synopsis:

`apropos` *keyword*

Arguments:

keyword	Any string.

Description:

The `apropos` command prints documentation which contains a match for *keyword*. If the command fails to find anything appropriate, the `help` command is invoked.

Examples:

```
apropos redraw
```

See Also:

`help`

autoscale — Set whether the view and data limits should be rescaled.

Synopsis:

`autoscale` *option=value [, option=value]*

Arguments:

option	One of the names of an autoscale property. See table below.
value	A value associated with the matching *option*. See table below

Option	Value	Value Meaning	Default Value
`data`	`on, off`	When `on`, rescale the limits of data so that colors will remap. For example, if you have a `pc` plot and you change the variable, the limits will rescale. When `off`, hold the data limits constant, so that changing a variable or adding a new plot has no effect on the data limits.	`on`
`view`	`on, off`	When `on`, rescale the view according to the limits of the plot. When `off`, hold the specified view limits so the view will not change when new files are opened or new plots are added.	`on`

Description:

By default, MeshTV automatically rescales the view extents when a new plot is added, or a new file is opened. If you want the view to remain the same, set `view` to `off`. MeshTV also automatically rescales the data extents. For example, if you have a pseudocolor (`pc`) plot of density, and you change the variable to pressure, the data limits would rescale and the colors of the pc plot would remap. If you want the data limits to remain constant when variables are switched, or during animations, set `data` to `off`.

Examples:

```
autoscale data=on view=on
```

See Also:

```
addframe, animate, newplot, play, powerwall, setframe, stop
```

banner — Controls plotting of banners in a visualization window.

Synopsis:

```
banner name on|off option=value [,option=value, ...]
```

Arguments:

name	The name of the banner. This can be either “top” or “bottom”.
on | *off*	A boolean value indicating whether or not the banner is displayed.
option	A banner option. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`text`	banner text	The text string that is used in the banner.	`“”`
`location`	{real number | center, real number | center}	This field specifies where the banner appears in the visualization window. Note that the word “center” can be given in place of a real number.	{center, 0.96} or {center, 0.015}
`height`	real number greater than 0.	This field specifies the height of the banner as a percentage of the visualization window height.	`0.025`
`tc`	color	Color to use for banner. This is either a color name or a color index. (See APPENDIX C: Color Names.)	`fore-ground`

Description:

The banner command allows banners, which are text strings, to be displayed in a visualization window. Only two banners can be placed in a visualization window and they are called “top” and “bottom”. These banners are turned on or off by giving a boolean on or off flag after the name. The banner text is set using the text argument and has a maximum length of fifty characters. The banner can be positioned in the visualization window using the location argument. The location argument expects a list of two real numbers enclosed in curly braces. The real numbers given in the location argument

specify the location of the banner's bottom left corner. These numbers are expressed as a percentage of the visualization window's height. The word "center" can be used in place of a number in the location argument. If that is done, the banner is centered in the specified dimension. The height argument controls the height of the banner. The height is specified as a percentage of the visualization window's height. Finally, the banner's color is set using the tc argument which takes either a color name or a color index.

Examples:

```
# Turn on the top banner.
banner "top" on, text="My Banner", tc=red
# Turn on the bottom banner and reposition it.
banner "bottom" on, text="foo", location={center, 0.2}, \
height=0.05
```

See Also:

`legend, triad`

bg — Redefine the background color of a window.

Synopsis:

`bg` *red green blue*

Arguments:

red	The red component of the color. This is a floating point percentage between 0. and 1.
green	The green component of the color. This is a floating point percentage between 0. and 1.
blue	The blue component of the color. This is a floating point percentage between 0. and 1.

This command sets the background color for the active window. If all color values are between 0. and 1. then the active window's background color will be changed to the color specified by the red, green, blue arguments when the window is next redrawn. If one or more color values are not between 0. and 1., an error message will appear and the command will be ignored.

Examples:

```
bg 0.0 0.0 1.0  #Set background color to blue.
bg 1.0 1.0 1.0  #Set background color to white.
```

See Also:

`colordef, fg, invert`

block — Set plotting options for the block decomposition plot.

Synopsis:

`block` *option=value [,option=value, ...]*

Arguments:

option A plot option for the block plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`colors`	{block# color [block# color ...]}	A list of block numbers and color names/indices. (See APPENDIX C: Color Names.) This specifies the colors to use for individual blocks.	
`lc`	color, `off`.	Coloring method to use for plot. If the value is `off`, cycled coloring is used. If the value is a color name or color index, solid coloring is used. (See APPENDIX C: Color Names.)	`off`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`pop`	See `pop` option on page 124.		`off`
`ptsize`	positive real number	Number to be used as scale factor to change the size of the marker for points when point meshes are displayed.	`.05`

Option	Value	Value Meaning	Default Value
`separate`	`on, off`	Whether or not to separate the blocks in the `internal` and `wireframe` plots.	`on`
`type`	`filled, internal, wireframe`	The method to use when plotting the blocks.	`filled`
`var`	variable name	Name of variable to get mesh data from. If the name is default, then the first appropriate variable found in the current directory of the file will be used.	`default`

Description:

The `block` command sets the options used when generating a block decomposition plot, where each block (domain) is assigned a color based on its block number.

Note that this command does not change any existing block plots, only future ones.

Examples:

```
block type=wireframe, lt=3, ls=dash, separate=on
block var=mesh1
```

See Also:

`plot, blocks`

blocks — Set the active block numbers.

Synopsis:

```
blocks block# [, [begin:end | block#], ...]
blocks begin:end [, [begin:end | block#], ...]
blocks all
```

Arguments:

block#	An integer representing a block identifier.
begin	An integer number representing the beginning of a range.
end	An integer number representing the end of a range.
`all`	Flag indicating that all blocks should be active.

Description:

The `blocks` command takes a comma-separated list of block numbers and/or begin-end pairs, and makes those blocks active. Only data from active blocks will be plotted.

A block is an element of a multimesh. A multimesh consists of several blocks, each of which is a separate mesh in its own right. Each block has its own variables, materials, etc.

Successive `blocks` commands are not cumulative, so each subsequent `blocks` command turns off currently activated blocks and turns on the ones that have just been specified.

If both a `groups` command and a `blocks` command have been specified, only those blocks which are in the intersection of the two sets will be selected.

Examples:

```
blocks 1,3,5:10# Activate 1,3,5,6,7,8,9, and 10
blocks 44      # Activate just block 44
blocks all     # Activate all blocks
```

See Also:

`groups`

bnd — Set plotting options for the material boundary plot.

Synopsis:

`bnd` *option=value [,option=value, ...]*

Arguments:

option A plot option for the material boundary plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`colors`	{material# color [material# color ...]}	A list of material numbers and color names/indices. (See APPENDIX C: Color Names.) This specifies the colors used for individual materials.	
`lc`	color, `off`	Coloring method to use for plot. If value is `off`, cycled coloring is used. If the value is a color name or index, solid coloring is used. (See APPENDIX C: Color Names.)	`off`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`pop`	See `pop` option on page 124.		`off`

Option	Value	Value Meaning	Default Value
`var`	variable name	Name of variable to get material data from. If the name is default, then the first appropriate variable found in the current directory of the file will be used.	`default`

Description:

The `bnd` command sets the options used when generating a material boundary plot. Note that this command does not change any existing material boundary plots, only future ones.

Examples:

```
bnd lc=red, ls=dot
bnd var=myvar
```

See Also:

`materials, plot, smat, mir`

cale — Access the Cale aliases.

Synopsis:

```
cale
```

Arguments:

No arguments.

Description:

Type this command to access the Cale aliases that MeshTV supports. (See APPENDIX B: Cale aliases.)

Examples:

```
cale
```

See Also:

```
alias
```

cat — Print the contents of the given file.

Synopsis:

`cat` *filename*

Arguments:

filename	The name of the file to print.

Description:

The `cat` command operates like the cat command in UNIX™ C-Shell, except it only works on one file at a time. Use this command if you want to examine the contents of a short ASCII file without opening up an editor or another window. If you issue this command from within the GUI, the output will go to the Output window, otherwise it will just print to the shell window running MeshTV.

Examples:

```
cat myfile
cat /usr/people/sample/input.file
cat ~/myfile
```

See Also:

`copyatt, sh`

cd — Change the active directory within the current SILO input file.

Synopsis:

```
cd
cd path
```

Arguments:

path	A path indicating the directory to change into.

Description:

The `cd` command operates within a SILO data file in the same manner that the UNIX™ cd command operates within the UNIX™ file system. By default, the active directory is the root directory, indicated by a slash "`/`". By using the `ls` and `cd` commands, one can navigate through a SILO file. The `cd` command understands both absolute and relative pathnames, hence the path "`..`" means move up one directory level. Without an argument, `cd` changes back to the root directory.

Note that this command works on SILO files, not UNIX™ files. This means that the "~" operator in a path will not work.

Examples:

```
cd /
cd new_dir
cd ../old_dir
```

See Also:

`close, ls, open, pwd`

center3 — Reassign the center point in the active 3D window.

Synopsis:

```
center3 xnum ynum znum
center3 off
center3 pick
```

Arguments:

xnum	X position for the center point. A floating point number.
ynum	Y position for the center point. A floating point number.
znum	Z position for the center point. A floating point number.
`off`	Return the center point to the default position.
`pick`	Choose the center point by picking a point in the window.

Description:

This command reassigns the “center point” of the view. In other words, it picks the point toward which the “camera” or “eye” is looking. This is particularly useful before a `zoomf3` to direct the eye to the area of interest before the area is enlarged.

When the pick option is used the cursor shape will be a plus sign. You can then press and release any mouse button over the point you want for the new center of rotation. The new center of rotation will correspond to the x, y, and z position of the point on the front surface under the mouse. The visualization window will go into navigate mode after the new point is selected.

Examples:

```
center3 1.5, 2.0, 3.5
center3 off
center3 pick
```

See Also:

`winset, zoomf3`

clear — Remove all plots from the visualization window(s).

Synopsis:

```
clear
clear all
```

Arguments:

`all`	Flag indicating that all visualization windows should be cleared.

Description:

The `clear` command clears (removes) all plots from the active window, or from all windows if the `all` argument is provided. In MeshTV, all plot requests are drawn on top of any existing plots. Only an explicit `clear` command or `newplot` command will clear the window.

Note that when the `-nowin` option is used on the command line for production mode, there are no windows per se. In this case, each successive plot command overlays the previous one, and when a `savewin` or `printwin` command is issued, all plots are output. In this case, the `clear` command can be used to create separate plots. Without it, plots made in sequence will be merged together.

Also note that the command `clear pick` will clear the pick point markers.

Examples:

```
clear
clear all
```

See Also:

`plot, newplot, redraw, winset`

close — Close the SILO file associated with the active visualization window.

Synopsis:

```
close
```

Arguments:

No arguments.

Description:

`close` closes the active window's SILO file. Note that issuing another `open` or `addframe` command also closes the currently open file.

Examples:

```
close
```

See Also:

```
open, winset
```

colordef — Redefine a color in the MeshTV standard color table.

Synopsis:

`colordef` *color_number (red green blue [scale [cname]]) | cname*

Arguments:

color_number	An integer from 1 to 30 which indicates which color to replace in the MeshTV standard color table. (See APPENDIX C: Color Names.) There are 9 colors in the table that have predefined names, but other colors can be referenced using their color number.
red	The red component of the color. This is a floating point percentage between 0. and 1.If any numbers are larger than 1, then they are scaled relative to the largest of the three numbers in the trio such that the largest number in the trio becomes one. If the scale value is supplied and is greater than the largest number in the trio, then the numbers in the trio are divided by the scale.
green	The green component of the color. This is a floating point percentage between 0. and 1.If any numbers are larger than 1, then they are scaled relative to the largest of the three numbers in the trio such that the largest number in the trio becomes one. If the scale value is supplied and is greater than the largest number in the trio, then the numbers in the trio are divided by the scale.
blue	The blue component of the color. This is a floating point percentage between 0. and 1. If any numbers are larger than 1, then they are scaled relative to the largest of the three numbers in the trio such that the largest number in the trio becomes one. If the scale value is supplied and is greater than the largest number in the trio, then the numbers in the trio are divided by the scale.
scale	An optional numeric value used to scale the numbers in the rgb trio if any of the numbers are greater than one. If the scale value is not supplied then the numbers in the rgb trio are scaled relative to the largest of the three numbers in the trio. This value must be provided if the red, green, blue components have been specified and the cname argument is provided.
cname	The name of the color to define or look up.

MeshTV provides 9 standard named colors. These can be replaced via the `colordef` command. To create a color via an existing X Windows color name, omit all arguments except color_index and cname.

This command defines a color to be used for block, material (`bnd`), contour (`iso`), `mesh`, filled material (`smat`), and `vec` plots. The color is comprised of red, green, and blue

components, and these are specified via a floating point number between 0 and 1. Numbers larger than one are scaled so that the largest value becomes one if the scale is not specified. This means that `“colordef 1 0 0.5 1”` is identical to `“colordef 1 0 10 20”`.

This command can be used to create custom color names that can be used with any commands that expect color names. To create a custom color name, all arguments must be specified. Colors can be redefined. Excluding the standard named colors, color names are only recognized in the window in which they were defined.

This command can be used to lookup colors in a list of defined colors. To find a list of defined colors available to MeshTV, look for the file: rgb.txt in the directory in which MeshTV was installed. If this command is being used to lookup a defined color, only the color_number and cname arguments are needed.

Note that if you are using the GUI and you issue this command in the command line interface window, the colors in the palette editor will no longer match the colors that you’ve specified in the command line interface window.

Examples:

```
#Set color 1 to aquamarine.
colordef 1 0.0 0.5 1.0
#Color 2 is the same aquamarine.
colordef 2 0 10 20

# Set color 30’s rgb values and create the name “purple”
colordef 30 0.5 0. 1. 1. purple

# Set color 1 to PapayaWhip
colordef 1 PapayaWhip
colordef 1 “papaya whip”
```

See Also:

`bnd, iso, mesh, smat, vec`

copyatt — Copy attributes from one window to another.

Synopsis:

`copyatt` *from_id to_id kind [kind...]*

Arguments:

from_id	The identifer of the window from which the attributes will be copied. It is an integer ranging between 1 and 16, and the particular window reference in the command must already exist.
to_id	The identifer of the window to which the attributes will be copied. It is an integer ranging between 1 and 16, and the particular window reference in the command must already exist.
kind	This is a string detailing what kind of attributes to copy. The following strings are valid: `all`, `annotation`, `lighting`, `palette`, `view`. If you specify '`all`', then annotation, lighting, palette, and view attributes will be copied. You must have at least one string in your command, but you can also specify multiple strings.

Description:

Copy attributes from one window, specified by *from_id* to another window, specified by *to_id*. To indicate which attributes to copy, include one or more of the following strings: `all`, `annotation`, `lighting`, `palette`, `view`. If you specify `all`, then annotation, lighting, palette, and view attributes will all be copied.

Examples:

```
copyatt 2 1 view    #Copy view attributes from win. 2 to 1
copyatt 4 2 all     #Copy all attr. from window 4 to 2
copyatt 3 4 view palette
```

See Also:

`show`

csh — Execute an operating system (shell) command.

Synopsis:

`csh` *command*
`csh` *“command args”*

Arguments:

command	Any shell command.
args	Arguments to the shell command.

Description:

The `csh` command is an escape to the operating system from within MeshTV. With it, you can enter any command which the operating system understands. Note that if the command has arguments, special characters (like ‘`*`’, ‘`/`’, ‘`&`’), or multiple commands separated by semi-colons, then the entire string must be quoted.

Use this command if you want to issue a UNIX™ command from within MeshTV.

Note that this command is identical to the `sh` command.

Examples:

```
csh “ls -l”
csh “emacs my-command-file &”
csh “rm junk*”
```

See Also:

`listdb, sh`

ct — Set the pseudocolor and surface plot color tables for the active window.

Synopsis:

`ct` *ct_name*

Arguments:

ct_name Either the name of a built-in color table, or the name of a palette file on disk. See the table below for built-in color tables.

Built-In Color Table Names	Description
`caleblack`	A reversed rainbow that starts with black.
`calewhite`	A reversed rainbow that starts with white.
`contoured`	A simple striped color palette containing four bands.
`gray`	A white to black palette.
`hot`	A hot to cold color palette, with blue representing low values and red high values. (Default)
`rainbow`	A rainbow palette.
`xray`	The inverse of the gray palette mentioned above.

Description:

The `ct` command changes the color table which is used by the pseudocolor (`pc`) command for the active window. The built-in color tables listed above are always available. In addition, MeshTV can read raw binary palette files produced by programs such as the National Center for Supercomputing Applications (NCSA) tools. The format for these binary files is 256 sets of 3 bytes where each respective byte in the trio represents red, green, and blue for each of the 256 colors.

Each window has its own color table, although on most systems you will only see one table in use at any one time. By changing the active window, issuing the `ct` command, and then the `reset` command, the palette will change.

Examples:

```
ct gray
ct /usr/local/pals/tim.pal
```

See Also:

`defct, pc, redraw, winset`

curve — Set options for curve plots.

Synopsis:

`curve` *option*`=`*value [, option*`=`*value, ...]*

Arguments:

option A `curve` command option. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`label`	`on, off`	Whether or not to display identifying labels on a curve.	`on`
`lc`	line color	Line color. (See APPENDIX C: Color Names.) `off` means the line color will cycle through the MeshTV standard color table. A color index may be used in place of a color name.	`off`
`legend`	`on, off`	Whether or not a legend for the reference line should be displayed with the plot.	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`

Description:

The `curve` command provides controls for curve plots. Curve plots are most often generated by the `refl` and `ref` commands, or by placing MeshTV in the line-out mode and drawing reference lines directly in the graphics visualization window via the mouse. See `winmode` for a description of the line-out mode.

By default, MeshTV cycles through the line colors for the curve plot. If you want all plots to be the same color, you should set the line color on the command line.

Note that this command does not change any existing plots, only future ones.

```
curve label=off, ls=dash, legend=on
```

```
curve lc=red, lt=2
```

See Also:

```
dist, distline, ref, refl, winmode
```

defct — Creates or alters a color table definition.

Synopsis:

`defct` *colortablename option=value [, option=value, ...]*

Arguments:

colortablename The name of the colortable to create or alter.

option An attribute than can be modified. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`colors`	{position r g b, position r g b, [position r g b...]}	This is a list of floating point numbers in the range of 0.0 to 1.0 that represent color control points. Each control point consists of four floating point numbers: position, r,g,b. The position specifies where the control point lies on a line segment [0.0, 1.0]. The r, g, b values specify the red, green, blue components of the control point. There must be at least two control points given when providing this option to the defct command.	
`copy`	color table name	This is a string value that gives the name of a MeshTV standard color table or a color table that has been previously defined with the defct command. The color table specified with this arguments is copied over the colortable that is being modified.	

Option	Value	Value Meaning	Default Value
`equal`	`on, off`	This flag alters the color table so the position of the control points is only used to determine their order. The color table is then generated with the assumption that the control points are all equally spaced.	`off`
`reset`	`on, off`	This option exists to provide a way to reset the attributes of a MeshTV standard color table. This flag has no effect on a user-defined color table.	`off`
`reverse`	`on, off`	This flag reverses the order of control points so the colortable is generated in reverse. This is easier than changing the control points.	`off`
`smooth`	`on, off`	This flag turns on linear interpolation of colors between control points, which leads to a smooth looking color table.	`on`

Description:

The `defct` command defines a new color table or alters an existing color table. To define a new color table, all that must be provided is a unique color table name and a list of colors. If an existing color table name is given, the new attributes are assigned to the pre-existing color table. The standard MeshTV color tables can be modified with the `defct` command as though they were user-defined color tables. The main difference is that the *reset* option can be used with the MeshTV standard color tables to reset them to their default attributes.

Once a color table is defined or modified with the `defct` command, the `ct` command must be used in order to set the color table in the window. This applies even if the altered color table is already set in the window. Setting the color table again with the `ct` command causes the new color table definition to be used when generating colors.

All MeshTV windows share the same color table definitions so any changes made to a color table will eventually show up in all windows using a color table that has been modified.

Examples:

```
#Change the MeshTV hot color table to be equal & not smooth.
defct hot equal=on, smooth=off
```

```
#Create a new color table called magma.
defct magma colors={0 0 0 0 .25 1 0 0 .66 1 1 0 1 1 1 1}

#Change the new magma color table so it is reversed.
defct magma reverse=on

#Make the MeshTV hot color table use its default values.
defct hot reset=on
```

See Also:

ct, pc, redraw, winset

defined — A conditional statement.

Synopsis:

`defined` *symbol*

Arguments:

symbol	Any string.

Description:

The `defined` command returns a non-zero value if the symbol is a command, function, or alias, else it returns 0. The symbol must be quoted if it might be an alias. This command is used primarily with the `if` command when you are programming your own aliases. If the command is being processed in an init file, the quotes must be preceeded by a backslash.

Examples:

```
if (defined “isovar”) (progn (delete iso1) (plot isovar))

# The following example is read from an init file.
alias command=yes plv " \
     if (defined \”vec_ppp\”)                        \
          (error \”The vector plot is already plotted.\”) \
          (progn                                       \
               (delete_vec)                             \
               (alias \”vec_ppp\” yes)                  \
               (set_vec_symm)                           \
               (vec var=$1)                             \
               (plot_vec))                              \
";
```

See Also:

`alias, error, if, progn`

defvar — Define a new variable from existing variables.

Synopsis:

```
defvar varname=expression
defvar varname
defvar
```

Arguments:

varname — The name of the new variable being defined.

expression — An expression based on the operators and functions defined in the table below. In general, variables, other defvar variables, functions, operators, and constants can be used in *expression*.

A variable referenced within *expression* can reside anywhere within the currently opened SILO file or anywhere within another SILO file. Referencing a variable by its name alone indicates it resides in the current directory of the currently opened SILO file.
The complete syntax for referencing a variable is:

```
silo_pathname:var_pathname
```

`silo_pathname:` is optional. If specified, it is the UNIX™ pathname to the SILO file in which the variable resides. If `silo_pathname` contains the "`/`" character, then the entire pathname must be enclosed by single quotes so that "`/`" is not taken to be the division operator. The pathname can be relative (`../path`, for example), and ~*username* syntax is allowed.

`var_pathname` is the pathname of the variable within the SILO file. If `var_pathname` is just the name of the variable, then it is assumed to reside in the current directory of the SILO file. The current directory when `silo_pathname` is specified is the root or "`/`" directory. If `var_pathname` has the "`/`" character, then the entire pathname must be enclosed by single quotes so that "`/`" is

not taken to be the division operator. Because this is a SILO file path, *~username* syntax is not allowed.

Function or Operator	Meaning	Usage
`{}`	Associate a list of expressions into a single variable. Usually used for defining a vector variable.	`{`$expr_1$`,`$expr_2$`[,...]}` $expr_i$ can be an expression of variables, other defvar variables, and constants that evaluate to a mesh variable.
`()`	Associative parenthesis.	Group mathematical operations, which are described in the table entry following this one.
`+, -, *, /, ^`	Addition, subtraction, multiplication, division, and exponentiation operators.	*expr op expr* *expr* can be an expression of variables, other defvar variables, and constants. *op* is one of the operators specified in the Operator column.
`abs`	Absolute value.	`abs(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants.
`acos`	Arccosine	`acos(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a scalar value. `acos` returns the arccosine of the *expr* in radians. Note that *expr* must range between -1 and 1, inclusive. It is not defined outside this range.

Function or Operator	Meaning	Usage
`addvf`	Add a constant volume fraction to a material.	`addvf(`*mat, num, vf*`)` *mat* can be either a valid MeshTV material variable or a defvar expression which evaluates to one. *num* must be a valid material number in *mat*. *vf* is a volume fraction (typically in the range of 0 to 1) which will be added to the given material. This evaluates to a valid MeshTV material variable and can be used in subsequent material plots.
`asin`	Arcsine	`asin(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a scalar value. `asin` returns the arcsine of the *expr* in radians. Note that *expr* must range between -1 and 1, inclusive. It is not defined outside this range.

Function or Operator	Meaning	Usage
`aslice`	Arbitrary slice (slice through a plane that is not necessarily orthogonal to the axes.)	`aslice(`*expr*`, fff,fff,fff, [i]`) *expr* as in `oslice` The first set of three numbers defines an origin, the second set defines a normal to the plane, and the third set defines the "upaxis" vector. None of these vectors must be unit vectors. The upaxis vector is used only when the arbitrary slice is mapped to 2D, and it specifies the direction that will be at the top of the 2D visualization window after the slice is performed. The upaxis vector doesn't have to lie along the plane. The last argument is optional, and it specifies the dimension for the slice. The only legal values are 2 and 3. The default value is 2, which means slices are mapped to 2D. If you specify 3 so that slices are left in 3D, you are duplicating the functionary of the old `gslice` operator.
`atan`	Arctangent	`atan(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a scalar value. `atan` returns the arctangent of the *expr* in radians.
`comp`	Create a scalar variable which is a component of a vector variable.	`comp(`*vector, component-number*`)` *vector* is the name of a valid MeshTV vector. *component-number* is the 0 origin number of the component to extract.
`const`	Create a variable which is a constant scalar or vector defined on the nodes or zones of a mesh.	`const(`*mesh*`,`*value*`,[nodal|zonal])` *mesh* is the name of a valid MeshTV mesh. *value* is either a vector of the form `{`*x*`,`*y*`[,`*z*`]}` or a single scalar value. `nodal` puts one value at each node, and `zonal` centers the values in the zones. If this parameter is not specified, the default is to place scalars in the zones and vectors at the nodes.

Function or Operator	Meaning	Usage
`coord0`	Create a variable which is the extraction of the first coordinate array from the specified mesh. The first coordinate array is usually the x coordinate array.	`coord0(`*mesh*`)` *mesh* is the name of a valid MeshTV mesh.
`coord1`	Create a variable which is the extraction of the second coordinate array from the specified mesh. The second coordinate array is usually the y coordinate array.	`coord1(`*mesh*`)` *mesh* is the name of a valid MeshTV mesh.
`coord2`	Create a variable which is the extraction of the third coordinate array from the specified mesh. The third coordinate array is usually the z coordinate array.	`coord2(`*mesh*`)` *mesh* is the name of a valid MeshTV mesh.
`cos`	Cosine	`cos(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a scalar value. `cos` returns the cosine of the *expr* as given in radians.

Function or Operator	Meaning	Usage

`dispcoord`	This function displaces the coordinate arrays of the mesh specified by the first argument by the vector specified as the second argument.	`dispcoord(` *mesh*`,` *vector* `)`
		mesh is an expression of variables, other defvar variables, and constants that evaluates to a valid MeshTV mesh. *vector* is an expression of variables, other defvar variables, and constants that evaluates to a mesh variable defined over the mesh given as the first argument.
		Example:
		`dispcoord(`*mesh*`, {coord0(`*mesh*`), coord1(`*mesh*`)})`
		This example displaces each coordinate by the distance of the point from the origin.

`gslice`	This operator is included only for backward compatibility, and it will be removed in the future. To achieve the same plot, use the `aslice` operator described in this section. This is a general slice, which was also called a cutplane. (This was combined with the `aslice` operator in the 3.4 release of MeshTV.)	`gslice(`*expr*`,` *long*`,`*lat*`,`*rad*`)` *expr* as in `oslice` *lat* rotates the plane about the axis. It varies from -90 to +90 degrees. Specifying a positive angle rotates the plane in a clockwise fashion when looking down on the X axis. *long* rotates the plane about the Y axis and can vary in value from -180 to +180 degrees. Specifying a positive angle rotates the plane in a counterclockwise fashion when looking down on the Y axis. *rad* moves the plane in and out along the direction specified by *lat* and *long*. *rad* can vary from -100 to +100.

Function or Operator	Meaning	Usage
`insel`	Index selection (i.e., reduce the size of the variable using begin, end, skip)	`insel(`*expr, ibeg [*`:`*iend [*`:`*iskip]]* *[*`,`*jbeg[*`:`*jend[*`:`*jskip]]]* *[*`,`*kbeg[*`:`*kend[*`:`*kskip]]],* `[group=`*group* `| block=`*block*`])` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a quadmesh variable. The options which have "*beg*" in their name represent the beginning index. Options with "*end*" represent the ending index, and options with "*skip*" indicate the number by which to skip. X, Y, and Z are represented by the i, j, and k prefixes. The k prefix options only work in 3D plots. In a multiblock mesh, you may specify a block number. The indices will then be local to that block. In a mesh with multiple groups, you may instead specify a group number. The indices will then be local to that group (and global across that group's blocks).
`ln`	Natural log	`ln(`*expr*`)` as with `abs`
`log`	Base-10 log	`log(`*expr*`)` as with `abs`
`matsel`	Material select	`matsel`(*expr, material-numbers*) *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a mesh variable. *material-numbers* can be a comma-separated list of material numbers and/or ranges, where a range consists of a begin-end pair.

Function or Operator	Meaning	Usage
`matgrow`	Increase the physical size of a material	`matgrow(`*expr, material-number, distance*`)` *expr* evaluates to a material variable. *material-number* must be a single number for a material in *expr.* *distance* specifies a distance in terms of the physical units of the problem for the size increase.
`matvf`	Material volume fraction	`matvf(`*expr, material-numbers, [material-numbers . . .]*`)` *expr* evaluates to a material variable. *material-numbers* can be either a a range of numbers or the word `all`. A range is specified in the manner *beg[:end[:skip]]*, where *beg* is the lowest number, *end* is the highest, and *skip* specifies the amount by which to skip. Any quantity of *material-numbers* may be entered.

Function or Operator	Meaning	Usage

`oslice`	Orthogonal slice (slice perpendicular to one of the axes)	`oslice(`*expr*`,`
		`ix=`*i* `|` `iy=`*j* `|iz=`*k* `|`
		`ix=`*{i,j,k}* `|` `iy=`*{i,j,k}* `|iz=`*{i,j,k}* `|`
		`xx=`*x* `|` `yy=`*y* `|` `zz=`*z* `|`
		`px=`*xpct* `|` `py=`*ypct* `|` `pz=`*zpct*`,`
		`[group=`*group* `|` `block=`*block*`])`
		expr can be an expression of variables, other defvar variables, and constants that evaluate to a mesh variable.
		Slicing can be done based on a zone index (the `ix`, `iy`, and `iz` options), on a coordinate value (the `xx`, `yy`, and `zz` options), or by a percent value (the `px`, `py`, and `pz`.) Only one of these options is specified at a time, with the x, y, or z indicating the axis to slice along.
		In a multiblock mesh, you may specify a block number when slicing by a zone index. The index will then be local to that block.
		In a mesh with multiple groups, you may instead specify in which group the zone index should be chosen. The index will then be local to that group (and global across that group's blocks).

`point2ucd`	Triangulates a point mesh (or point variable) into a UCD mesh (or UCD variable)	`point2ucd(` *mesh* `)` *mesh* is an expression of variables, other defvar variables, and constants that evaluate to a two-dimensional point mesh or point variable
`reflect`	Reflect about one or more axes.	`reflect(`*expr*`, xmin, xmax, ymin, ymax, zmin, zmax)` *expr* as in `oslice`. One or more of the `xmin`, `xmax`, etc. must be specified. `zmin` and `zmax` are only used for 3D problems.

Function or Operator	Meaning	Usage
`resrad`	Adjust resolution using a Monte Carlo resampling with the given radius.	`resrad(`*expr, radius*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a quadmesh variable. The mesh over which *expr* is defined must be two-dimensional and rectilinear.
`segment`	Select a zone in the mesh, and show its neighbors. The number of neighbors to show is covered by the *layers* argument.	`segment(`*expr, zone, layer [, block]*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a mesh variable. The mesh over which *expr* is defined must be three-dimensional. *zone* is the zone number to use as the "seed zone." It is the zone from which neighboring zones will be calculated. *layer* in an integer that represents the level of zones from the seed zone. For example, if you want to see all the neighbors that share a vertex with the seed zone, *layer* would be 1. If you wanted to see all the neighbors that share vertices with the first layer, *layer* would be 2. *block* is an optional integer that represents which block the seed zone is in. If nothing is specified, no blocks are used.
`sin`	Sine	`sin(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a scalar value. `sin` returns the sine of the *expr* as given in radians.

Function or Operator	Meaning	Usage
`specmf`	Species mass fraction	`specmf(`*expr, material, species*`)` *-or-* `specmf(`*expr, material, species, use_vf*`)` *expr* evaluates to a species variable. *material* and *species* can each be a a range of material/species numbers or the word `all`. A range of values is specified in the manner *beg[:end[:skip]]*, where *beg* is the lowest number, *end* is the highest, and *skip* specifies the amount by which to skip. With more than one material specified, the first form of this function will weight the mass fractions from each material by that material's volume fraction in that zone. With a single material specified, it will use the species mass fractions as-is. For control over this decision, include the optional argument *use_vf*. With *use_vf* set to `true`, the function will always weight species mass fractions by their material's volume fraction in a zone. This allows correct summation of mass fractions from separate calls to `specmf`. To disable weighting, set this option to `false`.
`sqrt`	Square root	`sqrt(`*expr*`)` as with `abs`
`tan`	Tangent	`tan(`*expr*`)` *expr* can be an expression of variables, other defvar variables, and constants that evaluate to a scalar value. `tan` returns the tangent of the *expr* as given in radians.

Description:

The `defvar` command defines new variables which can then be used in the same manner as variables from the data file. With no arguments, `defvar` simply lists all defined variables from the current MeshTV session. With one argument, `defvar` prints the current definition (if any) of the given variable to the terminal.

Examples:

```
defvar          # Print all definitions
```

```
defvar myvar   # Print the definition of myvar

defvar dnew reflect(d,xmin,ymin)
defvar pnew oslice(p,xx=10.)
defvar wow insel(reflect(oslice(d*(u^2 +
     v^2),py=50.),xmin),1:150:2, 1:75)

defvar velocity {u,v}
defvar momentum {u,v}*mass
defvar momentum {u*mass,v*mass}# Same as the previous line

defvar sum a + (b - c)
defvar factor 3.1415926/180.
defvar x factor*sqrt(u^2 + v^2) + .25

defvar displace dispcoord(quadmesh2d, {coord0(quadmesh2d),
     coord1(quadmesh2d)})

defvar spec1mat5 specmf(Spec,5,1)
defvar spec1and3 specmf(Spec,all,1:3:2)
defvar alloxygen specmf(Spec,5,6,true)+specmf(Spec,8,2,true)

defvar dp1 p - “/quad/p”
defvar dp2 p - “abc.silo:p”
defvar dp3 p - “/dir1/abc.silo:p”
```

See Also:

```
pickzone
```

delete — Delete plots.

Synopsis:

`delete` *plot_name [, plot_name, ...]*

Arguments:

plot_name	A plot name, like `smat` or `bnd`, followed by a number which indicates the position in the list of plots. For example, if there are two `smat` plots, these are referenced as `smat1` and `smat2`. If a single plot of a given type exists, it can either be referenced with the number appended, as in `bnd1`, or without the number appended. Plot types are: `bnd`, `iso`, `label`, `mesh`, `pc`, `pop`, `ribbon`, `smat`, `stream`, `surf`, and `vec`.

Description:

Delete plots of the given plot type. Note that if you want to delete multiple plots of the same type, you must either list the higher numbered plot first, or you must issue the same number. For example, if you have 3 mesh plots, you can type "`delete mesh3 mesh2 mesh1`", or "`delete mesh1 mesh1 mesh1`". This is because plots are renumbered after a deletion — if you have three mesh plots and you delete mesh2, then mesh3 will be renamed to mesh2.

Examples:

```
delete mesh3, mesh2, mesh1
delete bnd1
```

See Also:

`plot`

depthcue — Set the parameters for depth cueing.

Synopsis:

`depthcue` *option=value [, option=value, ...]*

Arguments:

option An option for depth cueing. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`type`	`off, fast, nice`	`off` disables all depth cueing. `fast` enables an approximate method, and `nice` enables an exact method. Note that for very complex datasets, `nice` may be faster when using the software renderer	`off`
`equation`	`linear, exp, exp2`	This sets the mathematical equation used for determining the color of an element. `linear` describes a linear falloff, `exp` is an exponential function, and `exp2` is the square of the exponential function	`exp2`
`fade`	positive real number	This is the amount of fade used in the `exp` and `exp2` equations. Typical values range from 0.5 to 10.	`3.0`
`start`	real number, 0<=`start`<=1 `start` < `end`	A distance from the viewer as a floating point percentage when the `linear` equation is used. No cueing is applied at a distance closer (lower) than this value.	`0.3`
`end`	real number, 0<=`end`<=1 `start` < `end`	A distance from the viewer as a floating point percentage when the `linear` equation is used. Past this distance, no objects can be seen.	`0.7`

Description:

Depth cueing fades objects into the background more if they are farther from the viewer. This enables objects closer to the viewer to be seen with more clarity, to hide distractions of objects farther away, and to give an impression of depth.

Depth cueing is disabled by default, but you can enable it by passing `fast` or `nice` as the `type`. `fast` looks good enough for most purposes, but `nice` is recommended for speed when using the software renderer on very complex datasets.

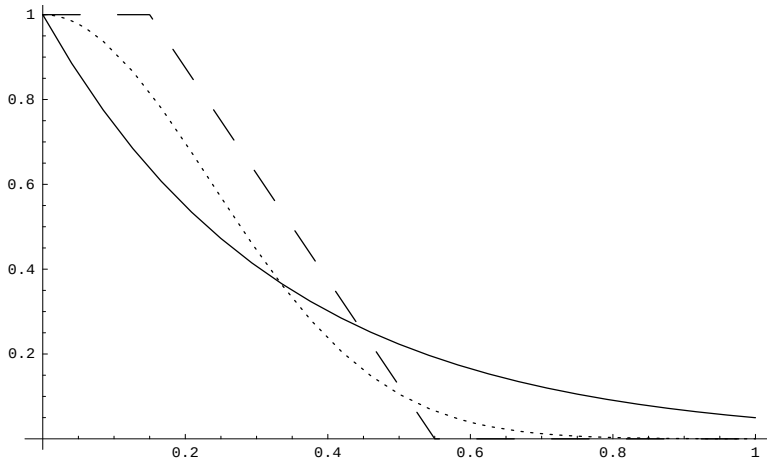
You can control both the style and degree of depth cueing by the equation and its parameters. The following equations determine the clarity of the object based on its distance from the viewer (z):

`linear:` $f = \frac{end - z}{end - start}$

`exp:` $f = e^{-(fade \cdot z)}$

`exp2:` $f = e^{-(fade \cdot z)^2}$

In the following diagram, the vertical axis represents the clarity of the drawn objects, and the horizontal axis represents distance from the viewer (0=closer). The dashed line represents the `linear` function, the solid line represents the `exp` function, and the dotted line shows the `exp2` function:



Using the `linear` equation and its `start` and `end` parameters, the depth cueing can be finely controlled. For a smoother look and simpler control, use either of the exponential equations, controlling the amount of cueing by the `fade` parameter. The `exp2` equation gives a steeper, sigmoid-like slope and is the default equation.

Examples:

```
depthcue type=fast  # enable depth cueing
depthcue equation=linear start=0.5 end=0.501
                    # sharply cut off everything halfway
depthcue equation=exp fade=1.5
                    # apply a little fade
depthcue equation=exp2 fade=5
                    # show only the closest objects
```

See Also:

dist — Set options for plotting a variable vs. distance along a line.

Synopsis:

`dist` *option*`=`*value [, option*`=`*value, ...]*

Arguments:

option A `dist` command option. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`lc`	line color	Line color. (See APPENDIX C: Color Names.) `off` means the line color will cycle through the MeshTV standard color table. A color index may be used in place of a color name.	`off`
`legend`	`on, off`	Whether or not a legend for the reference line should be displayed with the plot.	`off`
`logical`	`on, off`	When `on`, the distance plot will be created using logical index values, else it is created using coordinate values. This means that the numbers supplied by `distline` need to be either real or integer values as appropriate.	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`

Option	Value	Value Meaning	Default Value
`outwin`	`1 <=` integer `<= 16` or `-1`	Set the visualization window into which to display the distance plot curves. This must be different from the active window. If you specify -1 as the output window, you are indicating you want MeshTV to first try to use the first empty visuzliation window, or, if there are no empty windows, to open a new window to receive the curves.	`-1`
`repline`	`on, off`	If `on`, replace the last reference line with this one. If `off`, create a new reference line and add it to the display.	`off`
`repplot`	`on, off`	If `on`, update the curve associated with a reference line. If `off`, create a new curve anytime a reference plot is plotted or replotted.	`on`
`var`	variable name	Name of variable to plot.	`d`

Description:

The `dist` and `distline` commands provide shortcuts to the `ref` and `refl` commands, which are more general, but more verbose.

The `dist` command sets the options used when a MeshTV window is in the reference plot mode. By default, MeshTV will cycle through the line colors for the distance plot. If you want all plots to be the same color, you should set the line color on the command line. See `winmode` for a description of the ref mode.

The plot that displays in the visualization window is a plot of the chosen variable (set via the `var` option) as it varies along a line. The line is selected either by specifying endpoints, if using the `distline` command, or by drawing the line by clicking with the mouse in the active window while in the reference plot mode.

This command should only be issued if you have more than one visualization window created.

Examples:

```
dist var=d,outwin=2,logical=on
dist var=foo,outwin=4,lc=red
```

See Also:

`curve, distline, ref, refl, winmode, winnew, winnum`

distline — Plot a variable vs. distance along a line.

Synopsis:

`distline` *window_id* x_1 y_1 x_2 y_2

Arguments:

window_id	The identifer of the window which will contain the line, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.
x1,y1,x2,y2	The numbers are either the coordinates of the end-points of the line, in which case they are real (floating point) numbers, or they are logical indices into the mesh, in which case the numbers are integers.

Description:

A line with end-points `(x1,y1), (x2,y2)` is plotted to the window specified by *window_id*. A second curve is generated from the first, and it consists of all the values of a given variable at the points along the line. The variable is specified via the `dist` command, and the resultant values are plotted versus the distances of the points from the first end-point. This curve is plotted in the visualization window specified in the `dist` command.

Note that when the *logical* argument to the `dist` command equals `off`, *x1*, *y1*, *x2*, and *y2* must be real numbers (coordinates), and when *logical* equals `on`, these numbers must be integers.

Examples:

```
dist var=d, outwin=2 logical=off
distline 1 .3 .5 1.2 2.8 #Value curve goes to viz win 2
```

See Also:

`curve, dist, ref, refl, winnew, winnum`

echo — Print an informational message.

Synopsis:

`echo` *“message”*

Arguments:

message	A quoted string which contains the message to print.

Description:

This command prints an informational message to the standard error, which is usually the shell window from which the program is called. This is particularly useful when you are programming your own aliases, and you want to provide information. If the command is being processed in an init file, the quotes must be preceeded by a backslash.

Examples:

```
echo ”To use cale commands, type cale at the prompt.”
```

See Also:

`alias, defined, error, if, progn, warning`

end — Exit from the MeshTV session.

Synopsis:

```
end
```

Arguments:

No arguments.

Description:

The `end` command kills the MeshTV process and removes any visualization windows created during the session. A summary of the commands issued during the session is kept in the file "`%meshtv.log`" in the directory from which MeshTV was invoked. This log file can be used, after it is renamed, to run MeshTV with the same commands by using the `source` command from within MeshTV, or by invoking MeshTV at the UNIX™ prompt. For example, if you rename `%meshtv.log` to `file.log`, you would type the following "`meshtvx < file.log`". You must rename the file, since MeshTV will write to a file named `$meshtv.log`, and this would overwrite your commands.

Note that if you have issued an `foutput` command, and the information has yet to be written out, this command will force the information to be written.

This command is identical to the `quit` command.

Examples:

```
end
```

See Also:

`quit, source`

error — Print an error message.

Synopsis:

`error` *“message”*

Arguments:

message	A quoted string which contains the error message to print.

Description:

This command prints an error message to the standard error, which is usually the shell window from which the program is called. This is particularly useful when you are programming your own aliases, and you want to provide an error message. If the command is being processed in an init file, the quotes must be preceeded by a backslash.

Examples:

```
# The following example is read from an init file.
alias command=yes plv " \
     if (defined \”vec_ppp\”)                        \
          (error \”The vector plot is already plotted.\”) \
          (progn                                      \
               (delete_vec)                           \
               (alias \”vec_ppp\” yes)                \
               (set_vec_symm)                         \
               (vec var=$1)                           \
               (plot_vec))                            \
";
```

See Also:

`alias, defined, echo, if, progn, warning`

fg — Redefine the foreground color of a window.

Synopsis:

`fg` *red green blue*

Arguments:

red	The red component of the color. This is a floating point percentage between 0. and 1.
green	The green component of the color. This is a floating point percentage between 0. and 1.
blue	The blue component of the color. This is a floating point percentage between 0. and 1.

This command sets the foreground color for the active window. If all color values are between 0. and 1., then the active window's foreground color will be changed to the color specified by the red, green, blue arguments when the window is next redrawn. If one or more color values are not between 0. and 1., an error message will appear and the command will be ignored.

Examples:

```
fg 0.0 0.0 1.0  #Set foreground color to blue.
fg 0.0 0.0 0.0  #Set foreground color to black.
```

See Also:

`bg, colordef, invert`

flipxy — Flip the X and Y axis.

Synopsis:

```
flipxy on | off
```

Arguments:

`on`	Flip the X and Y axis.
`off`	Leave the X and Y axis in their standard positions.

Description:

The `flipxy` command sets the orientation of the X and Y axes. A value of “off” is the default, and this displays the X and Y axis in their standard orientation. A value of “on” places the X axis where the Y normally is, and the Y axis where the X axis normally is. When `flipxy` is on, the data is also flipped. (That is, this commands doesn’t only swap the axes.) This command applies only to 2D plots.

Examples:

```
flipxy on
```

See Also:

foutput — Send all subsequent plots to an output file.

Synopsis:

`foutput` *option=value [, option=value, ...]*

Arguments:

option An option for sending plots. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`banner`	banner string	This is a string to print to the top and bottom of a PostScript™ output file.	`""`
`rootname`	path name	Root name of output file family. Output file family members are named *pathname*0000.*filetype*, *pathname*0001.*filetype*, ...	`meshtv`
`type`	`ps, rgb, rps, tif`	The file format to save the image in. `ps` — Vector PostScript™, 2D plots only `rgb` — SGI's RGB format `rps` — Raster PostScript™ `tif` — TIFF format	`ps`
`xres`	positive integer	The resolution in the X direction. Only applies to `rgb`, `rps`, and `tif` formats. If you specify `xres`, you must also specify `yres`, and the two numbers should be equal to each other.	`1024`
`yres`	positive integer	The resolution in the Y direction. Only applies to `rgb`, `rps`, and `tif` formats. If you specify `yres`, you must also specify `xres`, and the two numbers should be equal to each other.	`1024`

Description:

The `foutput` command causes all subsequent plots to be sent to a file family in a format specified by `type`. Only one output file family can be active at a time—plots are directed to the file specified by the most recent invocation of `foutput`. Also, invocation of `foutput` causes a new file family to be created; therefore any existing family with the same root name is overwritten.

Plots from all visualization windows are written to the output file family when a command is invoked that acts like a frame advance for output files; that is, a command which causes the plots in that window to be replotted (`pan`, `rotation`, `roty`, `rotz`, `vp`, `wp`, `zoom3`) or causes the window to be cleared (`clear`, `end`, `newplot`, `newplot`, `powerwall`, `quit`). For example, invoking the `wp` command causes the plots to be first written to the output file and then replotted with new physical window limits. Invoking the `plot` command would not cause plots to be written since execution of the command adds a plot to the window rather than causing existing plots to be redrawn or cleared.

For reference line (distance) plots, two output file frames are generated, the first frame containing the reference lines, the second frame containing the reference curves.

Examples:

```
open prob1.silo
foutput rootname=prob1 # Create prob10000.ps
plot mesh
plot iso
wp 0 3 0 3     # Write mesh, iso frame.
clear          # Write mesh, iso frame with
               # new limits.
plot bnd
quit           # Write bnd frame.
```

See Also:

`clear, plot, wp`

fullframe — Set the aspect ratio of the graphics window.

Synopsis:

```
fullframe on | off
```

Arguments:

`on`	Force the viewport to be square.
`off`	Allow the viewport to conform to the aspect ratio of the data.

Description:

The `fullframe` command sets the value of the fullframe parameter to either “on” or “off”. A value of “on” forces the output viewport to be square, thereby allowing the aspect ratio of the data to be other than 1:1. A value of “off” is the default, and this causes the data aspect ratio to be 1:1. This command is useful when the image being examined is long and narrow, and the viewport is too narrow to see sufficient detail.

Examples:

```
fullframe on
```

See Also:

groups — Set the active group numbers.

Synopsis:

```
groups group# [, [begin:end | group#], ...]
groups begin:end [, [begin:end | group#], ...]
groups all
```

Arguments:

group#	An integer representing a group identifier.
begin	An integer number representing the beginning of a range.
end	An integer number representing the end of a range.
`all`	Flag indicating that all groups should be active.

Description:

The `groups` command takes a comma-separated list of group numbers and/or begin-end pairs, and makes those groups active. Only data from active groups will be plotted.

A group is a collection of blocks belonging to the same multimesh. The existence of a group can hide the details of its decomposition into blocks. For example, blocks within a group of structured meshes are usually logically connected and can be accessed using coherent logical indices. Separate groups are usually not connected in an orderly fashion and must be accessed in different ways.

Successive `groups` commands are not cumulative, so each subsequent `groups` command turns off currently activated groups and turns on the ones that have just been specified.

If both a `groups` command and a `blocks` command have been specified, only those blocks which are in the intersection of the two sets will be selected.

Examples:

```
groups 2,5:7   # Activate 2,5,6, and 7
groups 4       # Activate just group 4
groups all     # Activate all groups
```

See Also:

`blocks`

help — Print a command list or a brief summary of one command.

Synopsis:

```
help
help command
```

Arguments:

command	Any MeshTV command keyword.

Description:

The `help` command without arguments lists all MeshTV commands. When the `help` command is given a command keyword as an argument, it prints a brief summary of the requested command.

NOTE: This command has yet to be fully implemented.

Examples:

```
help redraw
help
```

See Also:

`apropos`

hide — Hide plots.

Synopsis:

`hide` *plot_name [, plot_name, ...]*

Arguments:

plot_name	A plot name, like `smat` or `bnd`, followed by a number which indicates the position in the list of plots. For example, if there are two smat plots, they are referenced as `smat1` and `smat2`. Plot types are: `bnd`, `iso`, `label`, `mesh`, `pc`, `pop`, `ribbon`, `smat`, `stream`, `surf`, and `vec`.

Description:

The `hide` command hides the specified plots. The plots can be unhidden using the `unhide` command. No action occurs when `hide` is applied to already hidden plots.

Examples:

```
hide pc2, mesh1
```

See Also:

`unhide`

history — Print the most recently entered MeshTV commands.

Synopsis:

`history` *[number]*

Arguments:

number	The number of commands to display.

Description:

The `history` command operates much like the UNIX™ C-Shell's history command. Without any arguments, `history` will show the last 20 commands given to MeshTV. The number of commands shown can be changed using the optional *number* argument.

To facilitate re-execution, the sequence number is printed beside the command. To reissue a command, use the `!` command, as in the UNIX™ C-Shell.

Examples:

```
history 10
history
```

See Also:

`!`

if — A conditional statement.

Synopsis:

`if` (*expression*) (*command*)
`if` (*expression*) (*command*) *[*(*command*)*...*(*command*)*]*

Arguments:

expression	Anything which can be evaluated. Commonly, this might be something like `defined “isovar”`, which is true if `isovar` is a command, function, or alias.
command	Any valid MeshTV command, or sequence of commands if you use the `progn` command

Description:

The `if` command takes an expression and a command to evaluate if the condition is true. Additional commands are evaluated if the expression is false. This is used when you are programming your own aliases.

Examples:

```
if (defined “isovar”) (plot isovar) (plot d)
if (defined “isovar”) (progn (delete iso1) (plot isovar))

# The following example is read from an init file.
alias command=yes plv " \
     if (defined \”vec_ppp\”)                        \
          (error \”The vector plot is already plotted.\”) \
          (progn                                      \
               (delete_vec)                           \
               (alias \”vec_ppp\” yes)                \
               (set_vec_symm)                         \
               (vec var=$1)                           \
               (plot_vec))                            \
";
```

See Also:

`alias, defined, error, progn`

inq — Inquire about the current SILO file attributes.

Synopsis:

```
inq
```

Arguments:

No arguments.

Description:

The `inq` command prints out the attributes of the objects in the current directory of the current SILO file. Among the information provided are mesh dimensions, number of materials, time, and cycle.

Examples:

```
inq
```

See Also:

```
show, SILO User's Guide
```

invert — `Swap background and foreground colors.`

`Synopsis:`

```
invert
```

Arguments:

No arguments.

Description:

The `invert` command changes the background color to the foreground color and changes the foreground color to the previous background color.

Examples:

```
invert
```

See Also:

`bg, fg, show`

iso — Set plotting options for the contour line/surface plot.

Synopsis:

`iso` *option=value [,option=value, ...]*

Arguments:

option A plot option for the contour plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`colors`	{contour# color [contour# color ...]}	A list of contour numbers and color names/indices. (See APPENDIX C: Color Names.) This specifies the colors for individual contours.	
`err`	`0.0` <= real <= `1.0`	Error tolerance for polygon decimation. This allows the user to remove polygons for faster display. The number indicates how far away from coplanar two polygons can be before they are no longer eligible to be collapsed into one polygon. A value of 0.0 will removes nothing, while a value of 1.0 will remove everything. This option applies only to 3D data.	`0.0`
`lc`	color or `off`	Coloring method to use for plot. If the value is `off`, cycled coloring is used. If the value is a color name or index, solid coloring is used. (See APPENDIX C: Color Names.)	`off`

Option	Value	Value Meaning	Default Value
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`level`	list of real numbers	Space-delimited set of numbers to use as actual iso level values. This conflicts with the `pct` and `nlevels` options, as explained in the Description section below.	`No default`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`max`	real number, `off`	Points above this maximum value will be ignored. `off` indicates the actual data maximum will be used.	`off`
`min`	real number, `off`	Points below this minimum value will be ignored. `off` indicates the actual data minimum will be used.	`off`
`nlevels`	positive integer	Integer number of levels, which will range uniformly from data min to data max of the chosen variable. This conflicts with the `level` and `pct` options, as explained in the Description section below.	`7`
`pct`	iso percent levels	Space-delimited set of percentages to use for calculating iso levels. This conflicts with the `level` and `nlevels` options, as explained in the Description section below.	`No default`

Option	Value	Value Meaning	Default Value
pop	See pop option on page 124.		off
scale	linear, log	Display the legend scale as a linear or log scale. This option only applies when the legend option is on.	linear
var	variable name	Name of variable to plot. If the name is default, then the first appropriate variable found in the current directory of the file will be used.	default

Description:

The iso command sets the options used when generating a contour (i.e., iso) plot. Note that this command does not change existing contour plots, only future ones. If plotting 2D data, the iso plot creates contour lines. If plotting 3D data, the iso plot creates contour surfaces.

The levels to draw are set by either the level, pct, or nlevels option. If more than one of these options is set, the last option "wins". The nelevels option is the default method for level selection.

To find the data value range of a variable, see the minmax command.

Examples:

```
iso lc=green,var=p, pct=10 25 50 75 90
iso level=0.5 1.0 2.0 5.0
```

See Also:

colordef, minmax, plot, pc

label — Set plotting options for labeling nodes and/or zones.

Synopsis:

`label` *option=value [, option=value, ...]*

Arguments:

option A plot option for the label plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`base`	real number > 1.0	This number controls how much labels “jump around” when you are zooming.	`2.0`
`hjust`	`l, r, c`	Horizontal justification of labels with respect to labelled objects. `l` — left-justified. `r` — right-justified. `c` — centered.	`c`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`

Option	Value	Value Meaning	Default Value
`mesh`	string	Name of variable or mesh whose values will be used as the labels. When this variable contains "default," MeshTV will use the first mesh it finds in the SILO file. When the string is the name of a variable (as opposed to a mesh variable), the `type` option (and thus the `node` and `zone` options) is ignored since the variable will print to nodes or zones as appropriate. This option is equivalent to the `var` option. (They used to be different, but both have been kept for backward compatibility.)	`default`
`nlabels`	`all`, positive integer	Set how many labels you want to see at a given time. For example, if the number is 100, you will see 100 labels in the image (if 100 exist) instead of seeing all the labels, which can lead to viewing problems if the number of labels gets to high.	`200`

Option	Value	Value Meaning	Default Value
`node`	`all`, one integer, two integers, three integers.	Label all nodes if "all" is the argument, else label the listed node. One integer lists a specific node. 2 numbers indicates (on collinear meshes) a logical x,y index, and 3 indicates (again only on collinear meshes) a logical x,y,z index. See the description for more detail. Note that the `type` option must have an "n" in it for this to actually plot, and the `var` option must not be used.	`all`
`pop`	See `pop` option on page 124.		`off`
`string`	string	The string with which to label nodes and zones if labeling a single element.	`*`
`tc`	text color	Color of labels. This can be specified as a color name or as a color index. (See APPENDIX C: Color Names.)	`fore-ground color`
`tht`	`0.0` < real number <= `1.0`	Height of labels.	`.025`

Option	Value	Value Meaning	Default Value
`type`	`n, z, nz, zn`	Type of mesh object to label. `n` — label nodes. `z` — label zones. `nz`, `zn` — label both. Note that this must be set for `node` and `zone` to take effect. Note also that this option is ignored if the `var` option is used since the variable will print to nodes or zones as appropriate for the variable type.	`n`

Option	Value	Value Meaning	Default Value
`var`	string	Name of variable or mesh whose values will be used as the labels. When this variable contains "default," MeshTV will use the first mesh it finds in the SILO file. When the string is the name of a variable (as opposed to a mesh variable), the `type` option (and thus the `node` and `zone` options) is ignored since the variable will print to nodes or zones as appropriate. This option is equivalent to the `mesh` option. (They used to be different, but both have been kept for backward compatibility.)	`No default`
`vjust`	`b, c, t`	Vertical justification of labels with respect to labelled object. `b` — bottom-justified. `c` — centered. `t` — top-justified.	`c`
`xoffset`	x offset from the point	Place the label at some fraction of the screen width away from the point, either negative or positive.	`0.0`

Option	Value	Value Meaning	Default Value
yoffset	y offset from the point	Place the label at some fraction of the screen width away from the point, either negative or positive.	0.0
zone	all, one integer, two integers, three integers.	Label all zones if “all” is the argument, else label the listed zones. One integer lists a specific zone. 2 numbers indicates (on collinear meshes) a logical x,y index, and 3 indicates a logical x,y,z index. See the Description section for more detail. Note that the type option must have a “z” in it for this to actually plot, and the var option must not be used.	No default

Description:

The label command sets the options used when generating a mesh label plot. Note that this command does not change existing label plots, only future ones.

The label command can label meshes at the nodes and/or zones, or it can label variables (like pressure and density) at the nodes or zones as appropriate. To label variables, set the var option to the name of the variable. If the var option isn’t specified, then node and/or zone numbers will be used for the plot, depending upon the value of the type option.

The type option indicates if we are labeling nodes, zones, or both. If “n” is chosen, then node must also have some value. It already has a default of “all”. In contrast, if “z” is chosen for the value of type, then zone must also be set to something, and it starts without a default. If type is set to “n”, then any value for zone is ignored, and if type is set to “z”, any value for node is ignored. If type is set to “nz” or “zn”, then both node and zone must have values. This option is ignored when var contains a variable.

All meshes supported by MeshTV can be labeled, but the node and zone options take different values depending on the mesh type. The values for node and zone options can

be the word "`all`" for all mesh types. The values can also be, for certain meshes, a set of numbers. All mesh types will allow one number. If one number is given, it is assumed to be the number of the node or zone to label, starting at whatever offset (0 or 1) that was given to the data file. Quadrilateral meshes may also be given 2 or 3 numbers, depending on the dimensionality (2D or 3D) of the mesh. In the 2D case, the numbers are interpreted as offsets into the mesh, so {1,2} means one node/zone in the "x" direction, and two in the "y" direction. The 3D case is the same, with an extension in the "z" direction. The actual direction might not be exactly parallel to the "x", "y", or "z" axes in non-colinear problems, but it is conceptually along those axes. Note that if 2 or 3 numbers are supplied to the node or zone options, the numbers must be inside braces, as shown in the first example below.

Examples:

```
label type=nz, mesh=mesh1, tc=red, node=all, zone={1,2}
label type=z, mesh=mesh1, tht=.02, hjust=l, vjust=b zone=2
label var=d, mesh=mesh2, nlabels=100
```

See Also:

```
mesh, plot
```

latitude — Set the 3D viewing parameter, latitude.

Synopsis:

`latitude` *number*

Arguments:

number	Any negative, zero, or positive real (floating point) angle in degrees.

Description:

The `latitude` command sets the viewing parameter which controls the angular distance positive or negative from the xy plane at z=0. When used in conjunction with the `longitude` command, this command allows you to set the viewing location around a 3D object. The viewing location is the place where your “eye” will be positioned.

Examples:

```
latitude 45
latitude -50.5
```

See Also:

`longitude, rotation, rotx, roty, rotz`

legend — Set the status and attributes of the various legends, which annotate plots.

Synopsis:

`legend` *option= value [, option = value, ...]*

Arguments:

option An option for the legend command. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`all`	`on, off`	Display all legends when on, else hide all legends. This is a quick way to toggle all legends on or off.	`off`
`axis2d`	`on, off`	Display 2D axis legends when on, else hide them.	`on`
`axis3d`	`on, off`	Display 3D axis legends when on, else hide them.	`on`
`axlab2d`	`x, y, xy, off`	Grid axes on which to place tick labels. `x` — tick labels along x-axis. `y` — tick labels along y-axis. `xy` — tick labels along both. `off` — turn off tick labels.	`xy`

Option	Value	Value Meaning	Default Value
`axlab3d`	`x, y, xy, xz, yz, xyz, off`	Grid axes on which to place tick labels. `x` — tick labels along x-axis. `y` — tick labels along y-axis. `z` — tick labels along z-axis. `xy` — tick labels along both. `xz` — tick labels along both. `yz` — tick labels along both. `xyz` — tick labels along all. `off` — turn off tick labels.	`xyz`
`db`	`on, off`	Display database legends when on, else hide them.	`on`
`gridax2d`	`x, y, xy, off`	Grid axes on which to draw major grid lines. `x` — grid lines along x-axis. `y` — grid lines along y-axis. `xy` — grid lines along both. `off` — turn off grid lines.	`off`

Option	Value	Value Meaning	Default Value
`gridax3d`	`x, y, xy, xz, yz, xyz, off`	Grid axes on which to draw major grid lines. `x` — grid lines along x-axis. `y` — grid lines along y-axis. `z` — grid lines along z-axis. `xy` — grid lines along both. `xz` — grid lines along both. `yz` — grid lines along both. `xyz` — grid lines along all. `off` — turn off grid lines.	`off`
`plot`	`on, off`	This option controls legends for the following plots: `bnd`, `iso`, `mesh`, `pc`, `pop`, `refl`, `ribbon`, `smat`, `stream`, `surf`, and `vec`.	`on`
`refl`	`on, off`	Display reference line plot legends when on, else hide them.	`off`
`time`	`on, off`	Display the current time, date, and user legend when on, else hide them.	`on`
`tickax2d`	`b, l, bl, bltr, off`	Grid axes on which to place ticks. `b` — bottom. `l` — left. `bl` — bottom left. `bltr` — bottom left top right. `off` — turn off tick labels.	`bl`

Option	Value	Value Meaning	Default Value
`tickax3d`	`x, y, xy, xz, yz, xyz, off`	Grid axes on which to place ticks. `x` — ticks along x-axis. `y` — ticks along y-axis. `z` — ticks along z-axis. `xy` — ticks along both. `xz` — ticks along both. `yz` — ticks along both. `xyz` — ticks along all. `off` — turn off tickmarks.	`xyz`
`tickloc2d`	`out, in, outin`	Location of ticks relative to axes.	`in`
`tickloc3d`	`out, in, outin`	Location of ticks relative to axes.	`in`
`verbose`	`on, off, diff`	Controls when to print database name, problem cycle, and problem time in individual plots. `on` — always plot `off` — never plot `diff` — Only show these for a plot when it is different from what is displayed at the top of the picture.	`diff`

Description:

The `legend` command controls which legends are plotted when annotating a display, and it controls the appearance of the legends. *legend_name* and *legend_att* settings can both be invoked within a single command. A *legend_name* value of `on` means plot the legend. (On is the default for all *legend_name* values.) Note that this command does not effect existing annotations, only future ones.

Examples:

```
legend all=off
legend all=on, iso=off, bnd=off
legend tickloc2d=out, tickax2d=bltr, surf=off
```

See Also:

`banner, bnd, iso, materials, mesh, pc, pop, refl, ribbon, smat, stream, surf, triad, vec`

lightsrc — Define lighting source properties for 3D plots.

Synopsis:

`lightsrc` *lightsrc_id [*`on`|`off`*]*
`lightsrc` *lightsrc_id [*`on`|`off`*]* `ambient` *color*
`lightsrc` *lightsrc_id [*`on`|`off`*]* `directional` *color direction*
`lightsrc` *lightsrc_id [*`on`|`off`*]* `eye` *color [direction]*

Arguments:

lightsrc_id	An integer ranging from 1 to 8 denoting which light source to define.
`on`	This specifies that the light source should be activated.
`off`	This specifies that the light source should be deactivated.
`ambient`	This light source type has no direction.
`directional`	This is a light source that stays fixed to the model.
`eye`	This is a light source that stays fixed to the viewer.
color	This can be either a color name (See APPENDIX C: Color Names.) or a red, green, blue triple.
direction	This is an x, y, z triple describing the direction in which the light source is facing.

Description:

Define light source properties for 3D plots. There are two lights (numbers 1 and 2) set by default. They are both directional and are facing <1.4, 1.0, 0.7> and <-1.4, -1.0, -0.7>. Both are white lights.

An ambient light is used to provide a background level of illumination so that unlit areas are not completely black. A directional light will always shine on the same spot on the model. An eye light is one which is fixed relative to the viewer's (window's) position in space. When no direction is specified for an eye light, it defaults to <0 0 -1>, which is directly into the screen.

Examples:

```
lightsrc 1 ambient 0.2 0.2 0.2
lightsrc 5 on
lightsrc 6 on directional white 1 0 0
lightsrc 7 directional 0.7 0.7 0.7 -1 0 0
```

See Also:

```
autoscale, colordef
```

listdb — List all SILO files in the specified directory.

Synopsis:

```
listdb
listdb pathname
```

Arguments:

pathname	Any absolute or relative directory path specification.

Description:

With no arguments, the `listdb` command lists all SILO files in the current working directory. If an argument is given, `listdb` lists all SILO files in the specified directory. Files of other types are not shown. Use this command to see which files are SILO files.

Examples:

```
listdb
listdb ../data
listdb ~/data
```

See Also:

`copyatt, sh`

lock — locks attributes of windows together

Synopsis:

`lock` *type window_id [window_id...]*

Arguments:

type	A string identifying the type of the lock. Currently only "`view`" is supported.
window_id	The identifier of the window to lock.

Description:

The lock command allows windows to be locked together so that when one updates, all other locked windows update with it.

Currently, only a window's view may be locked.

A list of locked windows for each type is kept. As windows are locked using the lock command, they are added to the list of locked windows. Windows can be removed from this list using the unlock command.

When a window is added to the list of locked windows, the shared attributes of the already-locked window are copied to the new window. In this way, a window may be "locked-to" an existing set of attributes.

When the attributes of one window in the list change, these changed attributes are copied to all of the windows in the list, causing the "lock" behavior.

Examples:

```
lock view 1
lock view 2 3 8
unlock view 3
```

See Also:

`copyatt, unlock`

longitude — Set the 3D viewing parameter, longitude.

Synopsis:

`longitude` *number*

Arguments:

number	Any negative, zero, or positive real (floating point) angle in degrees.

Description:

The `longitude` command sets the viewing parameter which controls the angular distance positive or negative from the zero meridian. When used in conjunction with the `latitude` command, this command allows you to set the viewing location around a 3D object. The viewing location is the place where your “eye” will be positioned.

Examples:

```
longitude 23.5
longitude -45
```

See Also:

`latitude, rotation, rotx, roty, rotz`

ls — List the contents of a directory within the current SILO file.

Synopsis:

`ls` *["options"] [pathname_list]*

Arguments:

options	A letter or string of letters preceded by a dash and encased in quotes. Default is "`-dmv`". See table below.
pathname_list	One or more pathnames to list. These can be relative or absolute.

Option	Meaning
`-a`	List everything in file. Activates the -c, -d, -m, -r, and -v flags.
`-c`	List curves.
`-d`	List directories.
`-l`	Produce a long list containing additional information.
`-m`	List meshes (both point, quad, and UCD mesh objects. (See SILO User's Guide, UCRL-MA-118751.).
`-r`	List material (region) information.
`-v`	List all quadvars, ucdvars, pointvars, and miscellaneous objects. See the SILO User's Guide for more information about these object types

Description:

The `ls` command operates on a SILO file in the same manner than the UNIX™ ls command operates on the UNIX™ file system. The user has control via the command options over which types of objects are listed. Without any options, `ls` lists directories, meshes, and mesh variables. The `ls` command understands both absolute and relative pathnames, hence the path '`..`' means to list the contents of the parent directory. Without any path arguments, `ls` lists the contents of the current SILO directory. Because this is a SILO file path and not a UNIX™ file path, *~username* syntax is not allowed.

When a file is opened, the active directory is the root directory, indicated by a slash "`/`". By using the `ls` and `cd` commands, the user can navigate through a SILO file.

Examples:

```
ls
ls "-v" ..
ls ../other_dir
```

See Also:

```
cd, close, open, pwd, SILO User's Guide
```

mapping — Set the 4x4 mapping matrix for the active display.

Synopsis:

mapping *matrixvalues*

Arguments:

matrixvalues	Sixteen real numbers that descrbe the elements of the 4x4 mapping matrix. Numbers are expected in row major order.

Description:

The `mapping` command allows the user to directly set the mapping matrix for the active display. The matrix specified by the mapping command is multiplied into the current mapping matrix to yield the new mapping matrix. This allows the user to specify the translation and scale of objects in a window. This command is useful in setting up a plot's appearance through a script and can be useful in movie generation through scripting.

Examples:

```
#Multiply the current mapping matrix by the identity matrix.
mapping 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1.

#Multiply the current mapping by this matrix to zoom the
#image by 1.5 times and do some translation.
mapping 1.5 0 0 0 0 1.5 0 0 0 0 1 0 -0.25 -0.25 0 1
```

See Also:

`pan, panf, rotation, zoom3, zoomf3`

materials — Set the active material numbers.

Synopsis:

`materials` *material [, [begin*`:`*end* | *material], ...]*
`materials` *begin*`:`*end [, [begin*`:`*end* | *material], ...]*
`materials all`

Arguments:

material	An integer representing a material identifier.
begin	An integer number representing the beginning of a range.
end	An integer number representing the end of a range.
`all`	Flag indicating that all materials should be active.

Description:

The `materials` command takes a comma-separated list of material numbers and/or ranges, where a range consists of a begin-end pair.

Examples:

```
materials 1,3,5:10# Activate 1,3,5,6,7,8,9, and 10
materials 44   # Activate just 44
materials all  # Activate all materials
```

See Also:

`matstat`

matspecies — Turn material species on and off.

Synopsis:

`matspecies` *material begin*`:`*end* | *species [, material begin*`:`*end* | *species...]*
`matspecies all`

Arguments:

material	An integer representing a material identifier.
begin	An integer number representing the beginning of a range of material species.
end	An integer number representing the end of a range of material species.
species	An integer number representing a material species.
`all`	Flag indicating that all materials and species should be active.

Description:

The `matspecies` command takes a comma-separated list of material numbers and/or ranges, where a range consists of a begin-end pair.

Examples:

```
matspecies 1 5:10# Activate species 5-10 for mat 1
matspecies 2 6 # Activate just species 6 for mat 2
matspecies all # Activate all material species
```

See Also:

`materials`

matstat — Print out statistics about a material.

Synopsis:

`matspecies` *material material-number* `[coord=[cyl|cart] [x=`*N*`|y=`*N*`|z=`*N*`]]`

Arguments:

material	An material variable.
material-number	An integer number representing a specific material.
coord	`cyl` for cylindrical coordinates and `cart` for cartesian coordinates This option is for two-dimensional meshes, and cylindrical is assumed.
x/y/z	A value specifying which axis (x or y) to revolve around and the position of the axis (usually 0), or which axis (z) to extrude along. This option is for two-dimensional meshes, and y=0 is the default.

Description:

The `matstat` command takes a material and material number, along with a coordinate specification, and prints out the surface area and volume of that material.

Examples:

```
matstat Material 2# Print stats for material 2 of Material
matstat Material 2 coord=cyl x=0# Same, but assume it is in
     cylindrical coordinates with x=0 the axis of revolution
```

See Also:

mesh — Set plotting options for the mesh plot.

Synopsis:

`mesh` *option=value [,option=value, ...]*

Arguments:

option A plot option for the mesh plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`err`	`0.0` <= real <= `1.0`	Error tolerance for mesh decimation. This allows the user to remove polygons for faster display. The number indicates how far away from coplanar two polygons can be before they are no longer eligible to be collapsed into one polygon. A value of 0.0 will touch nothing, while a value of 1.0 will remove everything. Note that this does not work for 2D UCD meshes.	`0.0`
`lc`	color	Line color or off. A color index may be used in place of a color name. (See APPENDIX C: Color Names.)	`off`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1` <= integer <= `4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`

Option	Value	Value Meaning	Default Value
`pop`	See `pop` option on page 124.		`off`
`ptsize`	positive real number	Number to be used as scale factor to change the size of the marker for points when point meshes are displayed.	`1.0`
`var`	variable name	Name of variable to get mesh data from. If the name is default, then the first appropriate variable found in the current directory of the file will be used.	`default`

Description:

The `mesh` command sets the options used when generating a mesh plot. Note that this command does not change any existing plots, only future ones.

Examples:

```
mesh ls=dash
mesh var=hydro, lc=red
```

See Also:

`defvar, plot`

mir — Set options for the material interface reconstruction algorithm.

Synopsis:

`mir` *option=value [,option=value, ...]*

Arguments:

option An option for the algorithm. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`algorithm`	`old, new`	Which algorithm to use. The old algorithm is an isosurface based algorithm, and it is less accurate. The new algorithm is a probability based algorithm which better handles clean zones and zones with more than two materials.	`new`
`subdiv`	`1,2,3`	The level of subdivision to use. `1` implies no subdivision, and the algorithm is fastest when level 1 is selected. 3 is the highest amount of subdivision and can better handle zones with higher numbers of materials, but it takes longer and can introduce minor oscillations in the material surface. Note that this only applies to 3D plots.	`1`
`iter`	nonnegative integer, typically between 0 and 20	This specifies the maximum number of iterations to use when attempting to make the volume fractions in the reconstruction represent the true volume fractions. Note that iteration can take a while in 3D, and that most of the accuracy is recovered after only a few iterations.	`0`
`vf`	real number from 0.0 to 1.0	This applies only to the old algorithm. If this number is less than 0.5, the material boundaries grow. If the number is more than 0.5, the material boundaries encompass less.	`0.501`

Description:

The `mir` command sets options which apply to future invocations of the material interface reconstruction algorithm. This algorithm is used when doing a filled or unfilled material boundary plot, or when doing material selection in 3D. Setting these options does not recalculate existing material interfaces, so a `replot` must be issued for each plot you wish to regenerate.

The `subdiv` and `iter` options apply only to the `new` algorithm, and the `vf` option applies only to the `old` algorithm.

Examples:

```
mir subdiv=3 iter=2
mir algorithm=old vf=0.4
```

See Also:

`replot, smat, bnd,` and `matsel` in the `defvar` command

minmax — Print a variable's minimum and maximum data values.

Synopsis:

`minmax` *varname*

Arguments:

varname	The name of the variable to examine.

Description:

The `minmax` command prints the minimum and maximum data values of the requested variable to MeshTV's text window. Use this when you want to set the range for the pseudocolor (`pc`) plot, or when setting contour (`iso`) levels. Note that if you are using blocks, this will print the minimum and maximum data values only for blocks which are active.

Examples:

```
minmax d
```

See Also:

`blocks, defvar, iso, ls, pc`

monitor — Print timing information for various plots.

Synopsis:

```
monitor
```

Arguments:

No arguments.

Description:

The `monitor` command prints timing (and other) information. In particular, it prints the number of lines and polygons generated, the time it took to render the plot, and the total time it took for the plot to both generate and render the plot. Please note that most plots do not supply the number of lines and polygons generated, so these two statistics are usually set to zero.

Each invocation of this command toggles the state. Originally, timing information isn't printed, so the first invocation of `monitor` turns on such printing. The second invocation turns it off again, and so forth.

Examples:

```
monitor
```

See Also:

navigate — Describes object appearance during rotations, pans, and zooms.

Synopsis:

`navigate` *type*

Arguments:

type	Either “`normal`” or “`bbox`”.

Description:

The `navigate` command controls the appearance of objects as they are interactively rotated, panned, or zoomed. When the navigation type is `normal`, operations (rotations, pans, and zooms) will involve the entire object with its existing appearance. When the navigation type is `bbox`, operations will involve a bounding box which encompasses the object. This allows operations to proceed much more quickly at the loss of visual detail. Once the operation stops, the object returns to its pre-operation appearance.

Examples:

```
navigate normal
navigate bbox
```

See Also:

newplot — Clear the screen when the next plot is drawn.

Synopsis:

```
newplot
```

Arguments:

No arguments.

Description:

The `newplot` command clears the current visualization window, but not until the next plot is ready to be drawn. Unlike the `clear` command, which clears the visualization window immediately, the `newplot` command is used to make a smooth transition from one plot to the next.

Examples:

```
newplot
```

See Also:

`clear`

nextframe — Plot the next frame in the current animation.

Synopsis:

```
nextframe
```

Arguments:

No arguments.

Description:

The `nextframe` command plots the next frame in the current animation. An animation is built through the use of the `open`, `addframe`, and `plot` commands.

Examples:

```
nextframe
```

See Also:

`addframe, animate, open, play, plot, powerwall, setframe, stop`

open — Open a SILO file for reading.

Synopsis:

`open` *filename*

Arguments:

filename	A filename, including either absolute or relative path specifications.

Description:

The `open` command opens a SILO file and associates it with the current MeshTV visualization window. All successive plots produced in the current visualization window will obtain data from the specified SILO file.

A SILO file is typically generated by a physics simulation application, and can contain physics meshes, variables associated with those meshes, subdirectories, curves, material data, and so on.

Previously opened SILO files are closed when a new `open` command is issued. To open a file without closing previously opened files, use the `addframe` command.

Examples:

```
open abc.silo
open /usr/people/sample/abc.silo
open ../data/abc.silo
open ~/data/abc.silo
```

See Also:

`addframe, close, listdb, replace, SILO User's Guide`

pan — Perform a relative pan viewing operation.

Synopsis:

`pan` *xpan ypan*

Arguments:

xpan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the X direction.
ypan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the Y direction.

Description:

The `pan` command performs a pan operation which is relative to the current view. That is, "`pan .1 .2`" would move the image 10% of the window width to the right and 20% of the window height upward. Successive `pan` commands with these same arguments would continue to pan the image. This is in contrast to the `panf` command, which performs an absolute pan operation.

Examples:

```
pan .5 0.
pan -.2 .2
```

See Also:

`center3, latitude, longitude, mapping, panf, zoom3, zoomf3`

panf — Perform an absolute pan viewing operation.

Synopsis:

`panf` *xpan ypan*

Arguments:

xpan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the X direction.
ypan	A real number whose value ranges from -1.0 to 1.0, indicating the amount to pan in the Y direction.

Description:

The `panf` (pan factor) command performs a pan operation which is absolute, based on the original image. For example, "`panf .1 .2`" would move the original image 10% of the window width to the right and 20% of the window height upward. Successive `panf` commands with the same arguments would have no effect. This is in contrast to the `pan` command, which performs a relative pan operation.

Examples:

```
panf .5 0.
panf -.2 .2
```

See Also:

`center3, latitude, longitude, mapping, pan, zoom3, zoomf3`

pause — Pause the execution of MeshTV and wait for a carriage-return.

Synopsis:

```
pause
```

Arguments:

No arguments.

Description:

The `pause` command pauses the execution of MeshTV. On most systems, MeshTV will wait until the user presses the Enter (Return) key. Use this to temporarily pause execution when you run MeshTV from a command input file.

Examples:

```
pause
```

See Also:

`source`

pc — Set plotting options for the pseudocolor contour plot.

Synopsis:

`pc` *option=value [,option=value, ...]*

Arguments:

option A plot option for the pseudocolor plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`max`	real number, `off`	This value is used for coloring the variable specified in `var`. Points at or above the maximum value will be shaded with the maximum value color. A value of “off” indicates the actual data maximum will be used.	`off`
`min`	real number, `off`	This value is used for coloring the variable specified in `var`. Points at or below the minimum value will be shaded with the minimum value color. A value of “off” indicates the actual data minimum will be used.	`off`
`pop`	See `pop` option on page 124.		`off`
`scale`	`linear, log, skew`	Display the legend scale as a linear, log, skew scale. This option only applies when the `legend` option is on.	`linear`

Option	Value	Value Meaning	Default Value
`skew`	real number > 0.	The skew factor that is used to map data values to color ranges. A value of 1. is equivalent to linear scaling. Values above or below 1 have the effect of mapping small data ranges to large color ranges to highlight differences in the data. This value is only used when the scale is set to skew.	`1.`
`type`	`natural`, `zone`, `zonesmooth`	Rendering type. `Zone` will color a zone a single color. `zonesmooth` assigns colors at the nodes and then interpolates the color between the nodes.	`natural`
`var`	variable name	Name of variable to plot.	`d`
`lit`	on, off	Toggles lighting	`on`

Description:

The `pc` command sets the options used when generating a pseudocolor contour plot. Note that this command does not change any existing plots, only future ones.

To determine the data value range, see the `minmax` command.

Examples:

```
pc var=d, type=zone
pc min=-10.5
pc min=1.0e-05, max=1.0e+5
```

See Also:

```
ct, iso, minmax, plot
```

perspective — Control perspective vs. orthographic projection.

Synopsis:

`perspective` *[*`on` | `off`*] [*`dscale=`*scale]*

Arguments:

`on`	Turn on perspective projection, which aids in seeing depth.
`off`	Turn off perspective projection and use orthographic projection. This kind of projection lacks depth cues.
`dscale`	This allows the user to scale the amount of perspective by changing the distance from the eye to the viewplane. The default is 2.0.

Description:

The `perspective` command toggles between perspective and orthogonal projection, and allows the user to set the amount of perspective. Issuing the command with no options, or with just the *dscale* option, automatically turns on perspective.

Examples:

```
perspective on dscale=2.5
perspective off
perspective dscale=10.0
perspective on
```

See Also:

pick — Set options for pick & query mode.

Synopsis:

`pick` *option=value [, option=value, ...]*

Arguments:

option	A plot option for the pick plot. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`tc`	text color	Text color. A color index may be used in place of a color name. (See APPENDIX C: Color Names.)	foreground
`var`	variable name	Name of variable to print information about.	`d`

Description:

The `pick` command sets the options for when MeshTV's visualization window is in pick mode. This mode allows the user to get information about points by selecting points on the screen using the mouse. Node numbers (for UCD meshes), zone indices (for quad meshes), and the value of the requested variable, materials, or matspecies will be printed to the text (shell) window. (If you issue this command from the GUI, the output will go to the MeshTV Output window.) The mouse selections must be made within the current visualization window, so this command cannot be used in production mode. See the `winmode` command for more info on the pick mode.

Picking operates on both 2D and 3D plots, but it works differently for each. In 2D, picking provides information about the data, whereas in 3D, picking allows you to interactively select a zone. This zone can later be used by `segment` (`See defvar command.`) as the seed zone. To do a pick & query for data on a 3D mesh, you must first slice the data with `aslice` or `oslice` (`See defvar command.`) to get a 2D plot, and then do a 2D pick & query. In 3D pick, the selected zone is marked by a "+" and the zone label. Only one zone is picked at a time, and once another zone is picked, the location of the previous zone is forgotten.

In 2D, each pick location is marked with a character that corresponds to one line of output. The default color for the pick character is the foreground color (which is usually white or black), but the color can be changed via the `tc` option.

This command also sets options for the `pickpt` command, which can be used without switching to pick mode.

Examples:

```
pick var=d
pick var=speed, tc=white
```

See Also:

```
defvar, pickpt, winmode, winset
```

pickpt — Pick information for a variable at a point.

Synopsis:

`pickpt` *window_id x y*

Arguments:

window_id	The identifer of the window containing the point, an integer ranging from 1 to the number of active windows. The maximum number is 16.
x, y	The coordinates of the point. These are floating point numbers.

Description:

`pickpt` prints the coordinates, node numbers (for UCD meshes), zone indices (for quad meshes), and the value of the requested variable, materials, or matspecies at the point (x,y) in graphics visualization window *window_id*. This information is printed to the text (shell) window, unless the command is issued from the GUI, in which case the information is printed to the MeshTV Output window. The command also labels the point. See the description of the `pick` command for details on point labeling. This command applies only to 2D picks.

Examples:

```
pick var=d, tc=white
pickpt 1 4.3 5.6
```

See Also:

`pick, winnew, winnum`

pickzone — Pick a seed zone for the segment operator.

Synopsis:

`pickzone` *window_id zonenum [blocknum]*

Arguments:

window_id	The identifer of the window containing the point, an integer ranging from 1 to the number of active windows. The maximum number is 16.
zonenum	LThe number of the zone picked as the seed zone. This is an integer.
blocknum	If the picked zone is in a dataset with blocks, you must supply the integer block number of the picked zone, else you can ignore this argument to the command.

Description:

`pickzone` prints information about the picked zone to the %meshtv.log file. It prints the window id number of the window in which the 3D pick happened, it prints the zone number of the picked zone, and it prints the block number of the picked zone if the problem dataset uses blocks. Placing this information into the log allows the user to rerun the log at a later date. If this information were not printed to the log file, the 3D pick wouldn't happen (since it's interactive) and the run would fail. This command applies only to 3D picks (a pick done in a 3D dataset when the mode is `pick`. See `winmode` command.)

Unlike the `pickpt` command, `pickzone` doesn't print data information, like the density, at the picked point. It is only used to set the initial zone for the segment operator.

Examples:

```
pickzone 1 522 8
```

See Also:

`defvar, pickpt, winmode`

play — Play the current animation.

Synopsis:

```
play
```

Arguments:

No arguments.

Description:

The `play` command commences playback of the current animation. Animations are created through the use of the `open`, `addframe`, and `plot` commands.

Examples:

```
play
```

See Also:

`addframe, animate, newplot, open, powerwall, rplay, setframe, stop`

plot — Generate a plot from the current data file.

Synopsis:

`plot` *plot_list*

Arguments:

plot_list A comma-separated list of plot names. See table below.

Plot Name	Plot Description
`block`	Color decomposition of a mutliblock mesh. Each block (domain) is assigned a color based on its block number.
`bnd`	Material boundary lines (for 2D data) or material boundary surfaces (for 3D data). Each line/surface is assigned a color based on its material number.
`iso`	Displays lines or surfaces of constant value. Contour lines are generated from 2D data and contour surfaces are generated from 3D data.
`label`	Displays node and/or zone numbers at the positions of a mesh's nodes and/or zones.
`mesh`	Displays the mesh.
`pc`	Displays a pseudocolored plot. Each zone or node is assigned a color, and the entire mesh is colored according to the current pseudocolor color table. (See `ct` command.) For 3D data, only the external faces are plotted.
`ref`	Displays data from a reference line which has already been plotted. `ref` and `refl` are closely related.
`refl`	Displays a reference line on a 2D data set. Data can be read from this line and displayed by the reference plot. `ref` and `refl` are closely related.
`ribbon`	Displays flow ribbons through a 3D vector field. Only works in 3D.

Plot Name	Plot Description
`smat`	Displays material boundaries in a solid-filled representation. Each zone is assigned a color based on its material number.
`stream`	Display the path of particles in a vector field.
`surf`	Displays a 3D surface of 2D data. The X and Y components are taken from the mesh, and the data variable is used as the Z component (height).
`vec`	Displays vectors based on the data variable. The length of the arrow corresponds to the magnitude of the vector, and the direction of the arrow corresponds to the direction of the vector.

Description:

The `plot` command generates the requested types of plots from the current data file. The new plots will be drawn on top of any existing plots already on the screen. If you want the plot to replace the contents of the window, first issue a `clear` command or a `newplot` command. For more information, see the command summary for each plot name.

Examples:

```
plot smat
plot iso, bnd
```

See Also:

`block, bnd, clear, delete, iso, label, mesh, newplot, pc, pop, ref, refl, reset, ribbon, smat, stream, surf, vec`

pop — Plot operation option to various commands.

Synopsis:

`pop=off`
`pop=`*`option(args)`*

Arguments:

`off` This is the default.

option A plot operation option. There are three choices, `clipp`, `reflectp`, and `symmetry`. The format for each option is:

`clipp([dist=`*`number`*`,]` *longitude, latitude, radius*`)`

`clipp2(`*origin, normal*`)`

`reflectp(xmin, xmax, ymin, ymax, zmin, zmax)`

`symmetry(ppp, npp, nnp, pnp, ppn, npn, nnn, pnn)`

args *Arguments to the matching operation option.*

For the `clipp` option, these are:

longitude Longitude relative to object

latitude Latitude relative to object

radius Radius relative to object

`dist=`*number*
Offset the clipped section by some floating point distance. This option can occur at any point in the argument string.

For the `clipp2` option, these are:

origin X,Y,Z triplet of the origin of a clipping plane

normal X,Y,Z triplet of the normal of the clipping plane

For the `reflectp` option, the values are:

`xmin`, `xmax` X line of reflection

`ymin`, `ymax` Y line of reflection

`zmin`, `zmax` Z line of reflection

`zmin` and `zmax` are only used for 3D plots. At least one argument is required, though all can be used, in any combination or order.

For the `symmetry` option, the values specify quadrants:

`ppp`	Positive X, Positive Y, Positive Z
`npp`	Negative X, Positive Y, Positive Z
`nnp`	Negative X, Negative Y, Positive Z
`pnp`	Positive X, Negative Y, Positive Z
`ppn`	Positive X, Positive Y, Negative Z
`npn`	Negative X, Positive Y, Negative Z
`nnn`	Negative X, Negative Y, Negative Z
`pnn`	Positive X, Negative Y, Negative Z

For 2D plots, use `pp`, `pn`, `np`, or `nn`.

Description:

Plot operations are operations that are performed on the geometry generated by the plots. Only one plot operator may be applied at a time.

The `clipp` operator allows the user to "prune" parts of a generated plot by defining a clipping plane. This is sometimes called the "erase" operator. The normal of the plane is specified with a longitude, latitude, and radius, and points which lie to that side of the plane will be plotted. Points on the opposite side of the plane will not be plotted unless the `dist` option is used. When the `dist` option is used, those points will be displaced along the inverse of the normal by the specified floating point value. Note that the `clipp` operator doesn't recreate data on the surface of the clipping plane. For example, if you apply the `clipp` operator to a pseudocolor plot, the resulting plot would be hollow at the point where the clipping plane intersected the plot.

The `clipp2` operator functions very similarly to the `clipp` command, although it has two major differences. The first is that it takes an origin/normal specification instead of a latitude/longitude/radius specification. Second, although both operators allow multiple planes to be specified in a single command, the `clipp2` operator performs intersections of the resulting half-spaces (while `clipp` performs unions).

The `reflectp` operator reflects the plot about the upper and lower boundaries along the various spacial directions. The user can independently control whether a reflection is done about a particular boundary. If a reflection is performed about both the upper and lower boundaries of a particular spacial direction, then the reflection about the upper boundary is done first. In addition, performing reflections about both the upper and lower boundaries results in 4 times the area/volume (not 3).

The `symmetry` operator is like the `reflectp` operator in some ways. For example you can achieve the same effect with both. For example, to have a mirror image of a mesh appear in the pn quadrant of a 2D mesh, you can say one of the following:

```
mesh pop=symmetry(pp, pn)
```

```
mesh pop=reflectp(ymin)
```

But the `symmetry` command can also plot just the mirror image in the `pn` quadrant by leaving out the `pp` specification:

```
mesh pop=symmetry(pn)
```

so you could get a mesh in the `pp` quadrant, a couple of pseudocolor plots in the `nn` and `pn` quadrants, and a filled boundary plot in the `np` quadrant. For an example, look at the file curv2d.silo (possibly located in /usr/local/meshtv/data; ask the person who installed MeshTV if you can't find it). Open it in meshtvx and type the following:

```
mesh var=curvemesh2d; plot mesh
pc var=d pop=symmetry(nmn); plot pc
pc var=u pop=symmetry(pn); plot pc
smat var=mat1 pop=symmetry(np); plot smat
```

This gives you an example of the power of the `symmetry` operator.

Examples:

```
bnd pop=clipp(0.0, 0.0, 1.0, dist=-4.0)
bnd pop=clipp2(0 0 0, 0 1 0)
iso pop=reflectp(xmin, xmax, ymin)
pc pop=symmetry(pp pn)
```

See Also:

`defvar`

powerwall — Controls a powerwall display.

Synopsis:

```
powerwall option=value [,option=value, ...]
```

Arguments:

option An option for controlling the powerwall. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`follow`	`on, off`	Whether or not to preserve window mapping, or to put plots only in the Powerwall active canvas. When follow is on, plots are only put in the Powerwall active canvas.	`on`
`layout`	`integer from 1 to 6`	This field specifies how the Powerwall is partitioned.	`1`
`mapwin`	`list of integers from 1 to 16`	This argument associates MeshTV windows with the active Powerwall canvas.	
`mode`	`on, off`	A flag indicating if the Powerwall is to be turned on or turned off.	`off`
`replot`	`on, off`	A flag indicating if the Powerwall should redraw plots when the Powerwall layout changes.	`off`
`winset`	`integer from 1 to 6`	This flag sets the Powerwall's active canvas. The maximum value can be no larger than the Powerwall's current layout.	`1`

Description:

The powerwall command controls a Powerwall and determines how MeshTV windows are displayed on it. A Powerwall is a large display that is composed of several smaller displays arranged in a tiled fashion. The mode flag turns the Powerwall on and off. The

layout flag determines how the Powerwall will be divided. Note that before the layout can be changed, the Powerwall must be turned off with the mode flag. The layout can then be changed and the Powerwall can be turned on in its new layout.

The remaining flags determine how a MeshTV window is put on the Powerwall. The winset flag sets the Powerwall's active canvas. The active canvas is the canvas on which MeshTV windows are displayed. The active canvas can only be set to values that are valid for the Powerwall's current layout. The follow flag determines whether or not all MeshTV windows will be displayed on the Powerwall's active canvas or if the windows will be displayed on the Powerwall canvas to which they are mapped. MeshTV windows are mapped to Powerwall canvases using mapwin. The mapwin argument associates the specified MeshTV window with the active powerwall canvas. Different MeshTV windows can be mapped to different powerwall canvases by first changing the Powerwall's active canvas with the winset flag and then using mapwin. Any MeshTV window can only be mapped to one Powerwall canvas at a time.

Examples:

```
# Turn on the powerwall in layout 3.
powerwall layout=3, mode=on
# Map MeshTV windows 1,2,3,4 to powerwall canvas 2.
powerwall follow=off, winset=2, mapwin={1 2 3 4}
# Make all MeshTV windows be drawn in powerwall canvas 1.
powerwall winset=1, follow=on
```

prevframe — Plot the previous frame in the current animation.

Synopsis:

```
prevframe
```

Arguments:

No arguments.

Description:

The `prevframe` command plots the previous frame in the current animation. An animation is built through the use of the `open`, `addframe`, and `plot` commands.

Examples:

```
prevframe
```

See Also:

`addframe, animate, open, newplot, play, rplay, setframe, stop`

printwin — Print the current visualization window on a network printer.

Synopsis:

`printwin`
`printwin` *option=value [,option=value, ...]*

Arguments:

option An option for printing images. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`banner`	banner string	This is a string to print to the top and bottom of a PostScript™ output file.	`""`
`capture`	`on, off`	Whether or not to get the image to save by doing a screen capture.	`on`
`command`	UNIX command	Use this field to specify a printer command, like `lp` or `lpr`. You can also modify this to add other printer options as long as the -P option remains the last on the line.	`lpr -P`
`printer`	printer name	The name of the printer to send the image to for printing.	`colorps`
`tiled`	`on, off`	A flag indicating whether or not to print all open graphics windows as a single image.	`off`
`type`	`ps, vps`	The type of the printer the image is being sent to for printing. `ps` — Raster PostScript™ `vps` — Vector PostScript™, 2D plots only	`ps`

Option	Value	Value Meaning	Default Value
`xres`	positive integer	The resolution in the X direction. Only used when `capture` is off, and only applies to the `ps` format. If you specify `xres`, you must also specify `yres`, and the two numbers should be equal to each other.	`1024`
`yres`	positive integer	The resolution in the Y direction. Only used when `capture` is off, and only applies to the `ps` format. If you specify `yres`, you must also specify `xres`, and the two numbers should be equal to each other.	`1024`

Description:

The `printwin` command without any options prints the contents of the current visualization window to the network printer. With options, the `printwin` command sets parameters for future window prints, *but does not actually print the window.*

The `ps` format corresponds to a raster image imbedded in a PostScript™ file and is resolution dependent. The `vps` format contains lines a polygons in a PostScript™ file and is resolution independent. Images with 3D graphics cannot be rendered using the `vps` format. When MeshTV detects a situation where the vps format cannot properly contain the image it will automatically switch to the `ps` format.

Note that if the `capture` flag is on, the picture will be “grabbed” via a screen capture. This means that if you open another window on top of your visualization window and then issue the `printwin` command, you will see the second window in your printed image. The default for `capture` is `on`, because this is the quickest way to get the picture. If the capture flag is off, the picture is generated internally at 24 bits at the specified resolution.

Examples:

```
printwin printer=color6
printwin tiled=on, printer=mycolor
printwin
```

See Also:

`savewin`

progn — A conditional statement.

Synopsis:

`progn (`*command*`)[(`*command*`)...(`*command*`)]`

Arguments:

command	Any valid MeshTV command.

Description:

The `progn` command allows multiple commands to be given in an `if` statement. This is used when you are programming your own aliases.

Examples:

```
if (defined “isovar”) (progn (delete iso1) (plot isovar))

# The following example is read from an init file.
alias command=yes plv " \
     if (defined \”vec_ppp\”)                        \
          (error \”The vector plot is already plotted.\”) \
          (progn                                      \
               (delete_vec)                           \
               (alias \”vec_ppp\” yes)                \
               (set_vec_symm)                         \
               (vec var=$1)                           \
               (plot_vec))                            \
";
```

See Also:

`alias, defined, error, if`

pwd — Print the name of the current directory in the active SILO file.

Synopsis:

```
pwd
```

Arguments:

No arguments.

Description:

The `pwd` command is analogous to the UNIX™ pwd command, except that it shows the current directory name within the active SILO file instead of within the UNIX™ file system.

Examples:

```
pwd
```

See Also:

`cd, close, copyatt, ls, open, sh`

quit — Exit from the MeshTV session.

Synopsis:

```
quit
```

Arguments:

No arguments.

Description:

The `quit` command kills the MeshTV process and removes any visualization windows created during the session. A summary of the commands issued during the session is kept in the file "`%meshtv.log`" in the directory from which MeshTV was invoked. This log file can be used, after it is renamed, to run MeshTV with the same commands by using the `source` command from within MeshTV, or by invoking MeshTV at the UNIX™ prompt. For example, if you rename `%meshtv.log` to `file.log`, you would type the following "`meshtvx < file.log`".You must rename the file, since MeshTV will try to write to a file named `$meshtv.log`, and this would overwrite your commands.

Note that if you have issued an `foutput` command, and the information has yet to be written out, this command will force the information to be written.

This command is identical to the `end` command.

Examples:

```
quit
```

See Also:

`end, source`

rect — Set options for plotting the integral of a variable within a rectangular box as a function of distance along the length.

Synopsis:

`rect` *option*=*value [, option*=*value, ...]*

Arguments:

option A rect plot option. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`lc`	line color	Line color. (See APPENDIX C: Color Names.) `off` means the line color will cycle through the MeshTV standard color table. A color index may be used in place of a color name.	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`outwin`	`1 <=` integer `<= 16`	Set the visualization window into which to display the distance plot. This must be different from the active window.	`2`
`var`	variable name	Name of variable to plot.	`d`

Description:

The `rect` command sets the options used when the `rectline` command is issued. By default, MeshTV will cycle through the line colors for the rect plot. If you want all plots to be the same color, you should set the line color on the command line.

The plot which displays in the visualization window is a plot of the chosen variable (set via the `var` option) as it varies along a line. The line is selected by specifying endpoints using the `rectline` command.

This command should only be issued if you have more than one visualization window created.

This command only works with rectilinear meshes.

Examples:

```
rect var=d,outwin=2
rect var=foo,outwin=4,lc=red
```

See Also:

```
rectline, winnew, winnum
```

rectline — Plot the integral of a variable within a rectangular box as a function of distance along the length.

Synopsis:

`rectline` *window_id* x_1 y_1 x_2 y_2

Arguments:

window_id	The identifer of the window containing the box, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.
x1,y1,x2,y2	The coordinates of two opposing points of a box. These are real (floating point) numbers.

Description:

A curve is generated from the values of a given variable, integrated in strips through the box. The variable is specified via the `rect` command, and the resultant values are plotted versus the distances along the box. This curve is plotted in the visualization window specified in the `rect` command.

Examples:

```
rect var=d, outwin=2
rectline 1 .3 .5 1.2 2.8 #Value curve goes to viz win 2
```

See Also:

`rect, winnew, winnum`

redraw — Redraw plots.

Synopsis:

```
redraw
redraw all
```

Arguments:

`all`	Redraw all visualization windows.

Description:

Redraw the plots in the active visualization window. `all` causes the plots in all windows to be redrawn. Calling `redraw` without arguments causes plots in the currently active visualization window to be redrawn.

Issue this command to redraw plots if they have become garbled on the screen. This command will also force some changes which apply to future plots (like changes to light sources or color tables) to apply to the current plot.

Examples:

```
redraw all
```

See Also:

ref — Set options for plotting a variable vs. distance along a line.

Synopsis:

`ref` *option=value [option=value]*

Arguments:

option A plot option for the `ref` plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`rl`	string	The name of the reference line plot from which to take data.	`a`
`var`	variable name	Name of the variable to use for the distance data.	`d`

Description:

A reference plot is the curve generated from data found in a reference line plot. A reference line plot (see the `refl` command) consists of a line drawn on a 2D data set. This line "selects" the nodes/zones used when a corresponding reference plot is done. For example, of you type "refl x1=1.0 y1=1.0 x2=2.0 y2=4.4" and then "plot refl", a reference line will draw to the specified window. If you then type "winnew; winset 2; plot ref", a reference plot will draw to the newly created (via `winnew`) and selected (via `winset`) window 2. This plot will display the value of "d", the default for the var option, along the reference line created when you typed "plot refl". As you can see, the two plot types, `ref` and `refl`, are inseparable.

Another example shows the value of the two shortcut commands, `dist` and `distline`. In this example, the data are plotted in window 1, and you have already created window 2. The goal is to draw a reference line in window 1, and see the results (a reference plot) in window 2. To accomplish this by using `ref` and `refl` commands, you would type the following:

```
refl x1=1.0 y1=1.0 x2=2.0 y2=2.0 name= “my_refl”
plot refl
winset 2
ref var=“d” rl=“my_refl”
plot ref
```

The equivalent results typed via the `dist` and `distline` commands would be:

```
dist var= “d” outwin=2
distline 1 1.0 1.0 2.0 2.0
```

You might wonder why we even have `ref` and `refl` since dist and distline seem much easier. They are easier, but they are also much less flexible, so for those users who want more control (including the ability to specify the reference line's endpoint via logical mesh indices rather than via coordinates), `ref` and `refl` are the commands to use.

Examples:

```
ref rl="line", var="p"
```

See Also:

`curve, dist, distline, plot, refl`

refl — Set options for plotting a variable vs. distance along a line.

Synopsis:

`refl` *option=value option=value*

Arguments:

option A plot option for the `refl` plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`ix1`	integer >= -1	The x index of the starting location for the reference line. When this value, `iy1`, `ix2`, and `iy2` are set to -1 and *logical* is `on`, then you can specify via coordinates (`x1`, `y1`, `x2`, `y2`) and the code will calculate the appropriate indices.	`-1`
`ix2`	integer >= -1	The x index of the ending location for the reference line. When this value, `ix1`, `iy1`, and `iy2` are set to -1 and *logical* is `on`, then you can specify via coordinates (`x1`, `y1`, `x2`, `y2`) and the code will calculate the appropriate indices.	`-1`
`iy1`	integer >= -1	The y index of the starting location for the reference line. When this value, `ix1`, `ix2`, and `iy2` are set to -1 and *logical* is `on`, then you can specify via coordinates (`x1`, `y1`, `x2`, `y2`) and the code will calculate the appropriate indices.	`-1`

Option	Value	Value Meaning	Default Value
`iy2`	integer >= -1	The y index of the ending location for the reference line. When this value, `ix1`, `iy1`, and `ix2` are set to -1 and *logical* is `on`, then you can specify via coordinates (`x1`, `y1`, `x2`, `y2`) and the code will calculate the appropriate indices.	`-1`
`lc`	line color	Line color. (See APPENDIX C: Color Names.) `off` means the line color will cycle through the MeshTV standard color table. A color index may be used in place of a color name.	`off`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`off`
`logical`	`on, off`	When `on`, the reference line will be created using locical index values, else it is created using coordinate values.	`off`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`name`	string	The name of a reference line.	`cycle`
`x1`	real number >= 0	The x coordinate of the starting endpoint for the reference line.	`0.0`
`x2`	real number >= 0	The x coordinate of the ending endpoint for the reference line.	`0.0`

Option	Value	Value Meaning	Default Value
`y1`	real number >= 0	The y coordinate of the starting endpoint for the reference line.	`0.0`
`y2`	real number >= 0	The y coordinate of the ending endpoint for the reference line.	`0.0`

Description:

A reference plot (see the `ref` command) is the curve generated from data found in a reference line plot. A reference line plot consists of a line drawn on a 2D data set. This line "selects" the nodes/zones used when a corresponding reference plot is done. For example, of you type "refl x1=1.0 y1=1.0 x2=2.0 y2=4.4" and then "plot refl", a reference line will draw to the specified window. If you then type "winnew; winset 2; plot ref", a reference plot will draw to the newly created (via `winnew`) and selected (via `winset`) window 2. This plot will display the value of "d", the default for the var option, along the reference line created when you typed "plot refl". As you can see, the two plot types, `ref` and `refl`, are inseparable.

Another example shows the value of the two shortcut commands, `dist` and `distline`. In this example, the data are plotted in window 1, and you have already created window 2. The goal is to draw a reference line in window 1, and see the results (a reference plot) in window 2. To accomplish this by using `ref` and `refl` commands, you would type the following:

```
refl x1=1.0 y1=1.0 x2=2.0 y2=2.0 name= "my_refl"
plot refl
winset 2
ref var="d" rl="my_refl"
plot ref
```

The equivalent results typed via the `dist` and `distline` commands would be:

```
dist var= "d" outwin=2
distline 1 1.0 1.0 2.0 2.0
```

You might wonder why we even have `ref` and `refl` since dist and distline seem much easier. They are easier, but they are also much less flexible, so for those users who want more control (including the ability to specify the reference line's endpoint via logical mesh indices rather than via coordinates), `ref` and `refl` are the commands to use.

Examples:

```
refl x1=1.0 y1 4.5 x2=165.2 y2=55.3 name="line"
```

See Also:

`curve, dist, distline, plot, ref`

reflect — Set the light reflectance method for the active visualization window.

Synopsis:

`reflect` *method*

Arguments:

method	The reflectance method: “`ambient`”, “`diffuse`”, “`specular`”, or “`none`”.

Description:

The reflectance method for the light source model is set to `method` for the active visualization window. The reflectance method determines the visual appearance of reflected light.

`ambient` adds reflected light from everywhere. It provides a kind of “background light”.

`diffuse` provides a matte or “dull” finish. This is a good method for objects which have a dull surface. When an object uses this method, it is scattering light equally in all directions.

`specular` produces a highlight on the object. This is a good method for shiny objects. When an object uses this method, light is reflected where the angle of incidence equals the angle of reflection, and this produces the highlight.

`none` ignores the reflectance method, or turns off a previously selected method.

Examples:

```
reflect diffuse
```

See Also:

`lightsrc`

renderer — Set global rendering attributes.

Synopsis:

`renderer` *option=value, [option=value]*

Arguments:

option	An option for the `renderer` command. See table below.
value	A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`dither`	`fast, nice`	Dithering method employed on pseudocolor displays. Dithering improves picture quality when color resources are limited.	`nice`
`offset`	`real number in the range [0.0, 0.2]`	Specifies how far mesh plots and wireframe blockplots should be drawn in front of other plots.	`0.002`
`optim2d`	`on, off`	When using the Software Renderer, setting this option to true causes all two dimensional plots to be drawn using the X11 renderer. This usually increases interactivity..	`off`

Description:

The `renderer` command sets global rendering attributes that are used for drawing in all MeshTV visualization windows. The `dither` option is used to improve image quality on pseudocolor displays where color resources are limited. If `dither` is set to fast, an ordered 4x4 dither is used. If `dither` is set to nice then Floyd-Steinberg error diffusion, which is the default, is used. The `offset` option is useful for 3d plots that include a mesh plot. The `offset` option determines how far in front of other plots the mesh plot is drawn. The `optim2d` option forces MeshTV to use the X11 renderer on 2d plots when using MeshTV with the Software Renderer. This can often improve interactivity.

Examples:

```
# Make MeshTV use the X11 renderer for 2d plots.
```

```
renderer optim2d=on

# Make MeshTV use Floyd-Steinberg dithering.
renderer dither=nice

# Set the mesh offset to a small number.
renderer offset=0.002
```

See Also:

```
redraw, replot
```

replace — Replace the currently open SILO file.

Synopsis:

`replace` *filename*

Arguments:

filename	A filename, including either absolute or relative path specifications.

Description:

The `replace` command opens a SILO file and associates it with the current MeshTV visualization window. All plots already displayed in that visualization window will be redrawn with data from the new file, keeping any attributes which have already been applied, such as data limits. In addition, all successive plots produced in the current visualization window will obtain data from the specified SILO file.

A SILO file is typically generated by a physics simulation application and can contain physics meshes, variables associated with those meshes, subdirectories, curves, material data, and so on.

Previously opened SILO files are closed when a `replace` command is issued. To open a file without closing previously opened files, use the `addframe` command.

Examples:

```
replace abc.silo
replace /usr/people/sample/abc.silo
replace ../data/abc.silo
replace ~/data/abc.silo
```

See Also:

`addframe, close, listdb, open, SILO User’s Guide`

replot — Replot the specified plots.

Synopsis:

`replot` *plot_name [, plot_name, ...]*

Arguments:

plot_name	A plot name, like `smat` or `bnd`, followed by a number which indicates the position in the list of plots. For example, if there are two `smat` plots, these are referenced as `smat1` and `smat2`. If a single plot of a given type exists, it can either be referenced with the number appended, as in `bnd1`, or without the number appended. Plot types are: `block`, `bnd`, `iso`, `label`, `mesh`, `pc`, `pop`, `ref`, `refl`, `ribbon`, `smat`, `stream`, `surf`, and `vec`.

Description:

Replot plots of the given plot type. This allows you to replot only one plot when you might have several more showing, thus reducing the amount of time needed to redraw the picture.

Examples:

```
replot mesh1, mesh2, smat
replot bnd1
```

See Also:

`plot`

reset — Reset options to their default values.

Synopsis:

`reset` *option*

Arguments:

option The name of option that should be reset.

Option Name	Meaning
`all`	All options which can be reset.
`bnd`	Boundary plot options.
`ct`	Color table (palette).
`curve`	Curve command options.
`iso`	Iso plot options.
`label`	Label plot options.
`legend`	Legend options.
`material`	Material options.
`mesh`	Mesh plot options.
`pc`	pseudocolor plot attributes.
`rect`	Rect command options.
`ribbon`	Ribbon plot options.
`ref`	Reference plot options.
`refl`	Reference line plot options.
`smat`	Solid-filled material plot options.
`stream`	Stream plot options.
`surf`	Surface plot options.
`vec`	Vector plot options.
`view`	Viewing parameters (e.g., window, up-vector, etc.).
`vp`	Viewport (e.g., xmin, xmax, ymin, ymax).

Option Name	Meaning
`win`	Options which modify the original position (rotation, panning, etc.) of the image.
`wp`	Physical window (e.g., xmin, xmax, ymin, ymax).

Description:

The `reset` command resets plotting, window, and display options back to their default values. This is a convenient way to reset plotting options (such as iso levels) or display parameters (such as the viewport) which have been changed.

Examples:

```
reset view
reset iso
reset all
```

See Also:

`bnd, ct, curve, iso, label, legend, materials, mesh, pc, pop, rect, ref, refl, ribbon, show, smat, stream, surf, vec, vp, wp`

ribbon — Set options for stream ribbons.

Synopsis:

`ribbon` *option=value [,option=value, ...]*

Arguments:

option A plot option for the ribbon plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`err`	real number `>= 0.0`	A scaled error value. An error of 0.01 means that variations on the order of 1/100 the size of the mesh are filtered out through line reduction. 0.0 indicates no filtering is done.	`0.0`
`density`	positive real number	The stream density. This is a scaled parameter. It relates to the “seed point density per unit area”; however, it is scaled such that a density of 1 results in approximately 200 streams. This is so the results are the same whether the mesh is defined in cm, mm, or any other scale.	`1.0`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`maxsteps`	positive integer	Number of integration steps taken per stream.	`50`

Option	Value	Value Meaning	Default Value

`method`	
	`euler`
	`rk2`
	`rk4`
	`eulera`
	`rk2a`
	`rk4a`
		Integration method for computing streams.
		1st order Euler integration
		2nd order Runge Kutta
		4th order Runge Kutta
		adaptive step 1st order
		adaptive step 2nd order
		adaptive step 4th order
			`rk2`

`numpoints`	positive integer	The number of streamlines to generate for each seed point. Two streamlines will result in a ribbon, while more than two will result in a tube.	`2`
`pop`	See `pop` option on page 124.		`off`
`step`	real number `>= 0.0`	Integration step size. This parameter is scaled according to the size of the mesh compared to the magnitudes of the vectors to allow the step size to give similar quality results in a wide variety of meshes. If using an adaptive method, the specified step is used only as an initial value and will be adjusted according to the curvature of the flow at each point.	`0.1`
`var`	variable name	Name of the vector variable. The variable defines pseudocoloring as well. A 4d vector is interpreted as {x_component, y_component, z_component, colorvar}.	`velocity`

Description:

Ribbons are an extension of streams for 3D meshes. Ribbons are generated by meshing a surface between multiple streamlines which are seeded near each other. This depicts the flow through a 3D problem.

Examples:

```
ribbon numpoints=3, method=”rk2a”, maxsteps=40
```

See Also:

```
pop, stream
```

rotation — Set the 4x4 rotation matrix for the active display.

Synopsis:

`rotation` *matrixvalues*

Arguments:

matrixvalues	Sixteen real numbers that describe the elements of a 4x4 rotation matrix. The numbers are expected in row major order.

Description:

The `rotation` command allows the user to directly set the rotation matrix for the active display. The matrix specified by the rotation command is multiplied into the current rotation matrix to yield the new rotation matrix. This allows the user to specify the rotations of objects in a window. This command is useful in setting up a plot's appearance through a script and can be useful in scripted movie generation.

Examples:

```
#Multiply current rotation by the identity matrix
rotation 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1.

#Multiply current rotation matrix by this matrix to rotate
#the plot by 45 degrees in X, then 45 degrees in Y.
rotation .71 0 -.71 0 .5 .71 .5 0 .5 -.71 .5 0 0 0 0 1
```

See Also:

`latitude, longitude, mapping, rotx, roty, rotz`

rotx — Rotate a 3D object about the screen's X-axis.

Synopsis:

`rotx` *angle [count]*

Arguments:

angle	A real number indicating the amount to rotate (in degrees).
count	Optional integer count parameter, indicating the number of times to rotate by `angle` degrees.

Description:

The `rotx` command allows the user to rotate a 3D object about the screen's X-axis. With the optional *count* parameter, one can rotate an object in multiple steps, thus getting a more complete picture of the object shape.

Examples:

```
rotx 45   # Rotate 45^o once
rotx 10 36# Rotate 10^o 36 times
```

See Also:

`latitude, longitude, rotation, roty, rotz`

roty — Rotate a 3D object about the screen's Y-axis.

Synopsis:

`roty` *angle [count]*

Arguments:

angle	A real number indicating the amount to rotate (in degrees).
count	Optional integer count parameter, indicating the number of times to rotate by `angle` degrees.

Description:

The `roty` command allows the user to rotate a 3D object about the screen's Y-axis. With the optional *count* parameter, one can rotate an object in multiple steps, thus getting a more complete picture of the object shape.

Examples:

```
roty 45   # Rotate 45^o once
roty 10 36# Rotate 10^o 36 times
```

See Also:

`latitude, longitude, rotation, rotx, rotz`

rotz — Rotate a 3D object about the screen's Z-axis.

Synopsis:

`rotz` *angle [count]*

Arguments:

angle	A real number indicating the amount to rotate (in degrees).
count	Optional integer count parameter, indicating the number of times to rotate by `angle` degrees.

Description:

The `rotz` command allows the user to rotate a 3D object about the screen's Z-axis. With the optional *count* parameter, one can rotate an object in multiple steps, thus getting a more complete picture of the object shape.

Examples:

```
rotz 45   # Rotate 45^o once
rotz 10 36# Rotate 10^o 36 times
```

See Also:

`latitude, longitude, rotation, rotx, roty`

rplay — Play the current animation backwards.

Synopsis:

```
rplay
```

Arguments:

No arguments.

Description:

The r`play` command commences playback of the current animation in reverse. Animations are created through the use of the `open`, `addframe`, and `plot` commands.

Examples:

```
rplay
```

See Also:

`addframe, animate, newplot, open, play, powerwall, setframe, stop`

savewin — Save the current visualization window into an image file.

Synopsis:

`savewin`
`savewin` *option=value [,option=value]*

Arguments:

option An option for saving images. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`banner`	banner string	This is a string to print to the top and bottom of a PostScript™ output file.	`""`
`capture`	`on, off`	Whether or not to get the image to save by doing a screen capture. Does not apply to STL files.	`on`
`family`	`on, off`	A flag indicating if the filenames should be familied.	`off`
`filename`	file name	The name (or rootname) of the file to save the image to. The proper extension (.ppm, .ps, .rgb, .tif) will be added onto the end of the filename.	`meshtv`
`tiled`	`on, off`	A flag indicating whether or not to save all open graphics windows as a single image. Does not apply to STL files.	`off`
`type`	`ppm, ps, rgb, rps, tif, stl`	The file format to save the image in. `ppm` ppm format `ps` PostScript™, 2D plots only `rgb` SGI's rgb format `rps` Raster PostScript™ `tif` Tiff format `stl` Stereolithography model	`ps`

Option	Value	Value Meaning	Default Value
`xres`	positive integer	The resolution in the X direction. Only used when `capture` is off, and only applies to `rgb`, `rps`, and `tif` formats. If you specify `xres`, you must also specify `yres`, and the two numbers should be equal to each other.	`1024`
`yres`	positive integer	The resolution in the Y direction. Only used when `capture` is off, and only applies to `rgb`, `rps`, and `tif` formats. If you specify `yres`, you must also specify `xres`, and the two numbers should be equal to each other.	`1024`

Description:

The `savewin` command without any options saves the contents of the current visualization window into the previously specified file. With options, the `savewin` command sets parameters for future window saves, *but does not actually save the window.*

If the capture flag is on, the picture will be “grabbed’ via a screen capture. This means that if you open another window on top of your visualization window before the save is done, you will see the second window in your saved file. This is the quickest method for saving pictures, which is why it is the default.

If the family flag is on, MeshTV will automatically append a four digit number to the end of the rootname, and automatically increment the number each time a `savewin` is done.

The types of file formats supported are ppm for Portable pixel map, `ps` for PostScript™, `rgb` for SGI’s rgb format, `rps` for raster PostScript™, and `tif` for Aldus Corporation’s tiff format.

Examples:

```
savewin filename=mypix, type=rgb
savewin tiled=on, family=on
savewin
```

See Also:

`printwin`

setenv — Set an environmental variable from within MeshTV.

Synopsis:

`setenv` *name command*

Arguments:

name	The name of the variable you're setting. The most useful one from MeshTV's point of view would be MESHTVHOME, since this variable needs to be set for MeshTV to find certain information, like the release notes.
command	Anything the UNIX™ command can take as the argument for the setenv comand. If you're setting MESHTVHOME, for example, *command* would be a path, like /usr/local/apps/meshtv/.

Description:

The `setenv` command allows you to define an environmental variable, just like the setenv command in UNIX™ does. This is particularly helpful if you run MeshTV and discover that MESHTVHOME was never set. You can set it without having to exit MeshTV first. Note that if *command* has special characters or spaces, then the entire string must be quoted.

Use this command if you want to set an environmental variable from within MeshTV.

Examples:

```
setenv MESHTVHOME "/usr/local/apps/meshtv/"
```

See Also:

`sh`

setframe — Set the current frame of an animation.

Synopsis:

`setframe` *frame_number*

Arguments:

frame_number	A frame number in the animation.

Description:

After an animation has been stopped, a specific frame in an animation can be selected via the `setframe` command. This frame will be displayed, and it becomes the current frame.

Examples:

```
setframe 0# Reset the animation.
setframe 10# Go to the 11th frame in the animation.
```

See Also:

`addframe, animate, newplot, play, powerwall, rplay, stop`

sh — Execute an operating system (shell) command.

Synopsis:

`sh` *command*
`sh` *“command args”*

Arguments:

command	Any shell command.
args	Arguments to the shell command.

Description:

The `sh` command is an escape to the operating system from within MeshTV. With it, you can enter any command which the operating system can understand. Note that if the command has arguments, special characters, or multiple commands separated by semi-colons, then the entire string must be quoted.

Use this command if you want to issue a UNIX™ command from within MeshTV.

Examples:

```
sh “ls -l”
sh “emacs my-command-file &”
sh “rm junk*”
```

See Also:

`copyatt, listdb, setenv`

show — Display the current settings for plotting options.

Synopsis:

`show` *option*

Arguments:

option An option name. See table below.

Option Name	Meaning
`block`	Show the block plot options.
`blocks`	Show the active block numbers.
`bnd`	Boundary plot options.
`colordef`	Show the rgb components of all colors.
`curve`	Curve command options.
`defct`	Color table attributes for all the defined color tables are displayed when "show defct" is executed. This includes color control points, color table name, smoothness, etc.
`foutput`	Display file output options.
`groups`	Lists which groups are active.
`iso`	Iso plot options.
`label`	Label plot options.
`legend`	Legend options.
`lightsrc`	Show the light source attributes.
`material`	Options for material numbers.
`matspecies`	Options for material species numbers in a plot.
`mesh`	Mesh plot options.
`pc`	pseudocolor plot options.
`pick`	Show options for pick and query mode.

Option Name	Meaning
`rect`	Rect plot options.
`ref`	Reference plot options.
`refl`	Reference line plot options.
`ribbon`	Ribbon plot options.
`smat`	Solid-filled material plot options.
`stream`	Stream plot options.
`surf`	Surface plot options.
`vec`	Vector plot options.
`view`	Viewing parameters (e.g., window, up-vector, etc.).

Description:

The `show` command prints the current values for the given option to the terminal. Use this to check the current variable or other option for a specific plot.

Examples:

```
show smat
show view
```

See Also:

`bnd, ct, curve, iso, label, legend, materials, mesh, pc, pop, rect, ref, refl, reset, ribbon, smat, stream, surf, vec`

simclose — Close an open simulation.

Synopsis:

`simclose` *[simname]*

Arguments:

simname	The name of a simulation, which follows the same rules as names for files. If this argument is omitted, MeshTV will close the currently active simulation.

Description:

The `simclose` command closes a remote computer simulation that was opened previously with the `simopen` command.

Examples:

```
simclose
simclose mycode
```

See Also:

`simcont, simopen, simpause, SILO User's Guide`

simcont — Continue a paused simulation.

Synopsis:

`simcont` *[simname]*

Arguments:

simname	The name of a simulation, which follows the same rules as those for files. If this argument is omitted, MeshTV will continue the currently active simulation.

Description:

The `simcont` command continues a remote computer simulation that was previously paused with the `simpause` command.

Examples:

```
simcont
simcont mycode
```

See Also:

`simclose, simopen, simpause, SILO User's Guide`

simopen — Open a remote computer simulation for reading.

Synopsis:

`simopen` *simname*

Arguments:

simname	A simulation name, which follows the same rules as those for files.

Description:

The `simopen` command opens a remote computer simulation which is running SILO's Scientific Data Exchange (SDX) package. MeshTV can read from the simulation just as it can read from a SILO file.

When a simulation is opened, it is implicitly paused. To continue the simulation, use the `simcont` command.

Examples:

```
simopen mycode
```

See Also:

`simclose, simcont, simpause, SILO User's Guide`

simpause — Pause execution of the current remote computer simulation.

Synopsis:

`simpause` *[simname]*

Arguments:

simname	The name of a simulation, which follows the same rules as those for files. If this argument is omitted, MeshTV will pause the currently active simulation.

Description:

The `simpause` command pauses execution of a remote computer simulation opened with the `simopen` command.

Examples:

```
simpause
simpause mycode
```

See Also:

`simclose, simcont, simopen, SILO User's Guide`

smat — Set plotting options for the solid-filled material boundary plot.

Synopsis:

`smat` *option=value [,option=value, ...]*

Arguments:

option A plot option for the material boundary plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`colors`	{material# color [material# color ...]}	A list of material numbers and color names/indices. (See APPENDIX C: Color Names.) This specifies the colors to use for individual materials.	
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`pop`	See `pop` option on page 124.		`off`
`type`	`all, clean`	Type of zones to plot. `all` will plot all materials, and `clean` will plot only clean zones (i.e., no zones with mixed materials.)	`all`
`var`	variable name	Name of variable to get material data from. If the name is default, then the first appropriate variable found in the current directory of the file will be used.	`default`

Description:

The `smat` command sets the options used when generating a solid-filled material boundary plot. Note that this command does not change any existing plots, only future ones.

Examples:

```
smat type=clean
smat var=myvar
```

See Also:

```
materials, bnd, plot, mir
```

source — Process a command input file.

Synopsis:

`source` *filename*

Arguments:

filename	The name of a file containing MeshTV commands to execute. The filename can include an absolute or relative path, though the "~" can not be used.

Description:

The `source` command operates like the UNIX™ C-Shell source command. It accepts a filename as input, then executes each line in that file as if a user had typed it in. Use this when you want to operate on a large set of files, but don't want to manually execute the plotting commands.

Two potential uses for a command input file are: (1) using MeshTV in a "batch" mode, where a large number of commands, files, or both are processed without user intervention; and (2) storing a series of commands which may be too long or complex for an alias, but which the user wants to execute frequently.

Examples:

```
source batch.file
```

Sample contents of 'batch.file':

```
plot pc# Assumes user already has opened a SILO file
winset 2
plot iso
winset 3
plot mesh
winset 4
plot vec
```

See Also:

`alias, copyatt, sh`

spinmode — Set rotational spin on or off.

Synopsis:

```
spinmode on | off
```

Arguments:

`on`	Objects which are rotated should continue to spin.
`off`	Rotated objects should stop spinning when they reach the indicated position.

Description:

The `spinmode` command indicates whether objects should continue to spin once the mouse is released. A value of “off” is the default, and this causes rotation to halt once the mouse button is released. A value of “on” allows rotation to continue even after the mouse is released. When this option is on and the object is spinning, you can stop the rotation by left clicking on the picture.

Examples:

```
spinmode on
```

See Also:

stereo — Set the stereo display mode.

Synopsis:

```
stereo on | off
```

Arguments:

`on`	Activate stereo mode.
`off`	Deactivate stereo mode. This is the default.

Description:

The `stereo` command sets the stereo viewing mode for the workstation. When “on”, the active window is enlarged to full-screen size, and a double set of images are created (one for the left eye, one for the right eye.) With the use of stereo viewers, the user can view images in 3-D.

NOTE - This option is available only on the SGI, and only on those machines with stereo viewers attached. Use of this command without the proper hardware attached will result in a “garbage” picture.

Examples:

```
stereo on
```

See Also:

stop — End the playback of the current animation.

Synopsis:

```
stop
```

Arguments:

No arguments.

Description:

The `stop` command stops the playback of the current animation. Resume the animation with the `play`, `newplot`, or `powerwall` commands.

Examples:

```
stop
```

See Also:

`addframe, animate, newplot, play, powerwall, rplay, setframe`

stream — Set options for the stream plot.

Synopsis:

`stream` *option=value [,option=value, ...]*

Arguments:

option A plot option for the stream plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`err`	real number `>= 0.0`	A scaled error value. An error of 0.01 means that variations on the order of 1/100 the size of the mesh are filtered out through line reduction. 0.0 means that no variations are filtered.	`0.0`
`density`	positive real number	The stream density. This is a scaled parameter. It relates to the “seed point density per unit area”, however it is scaled such that a density of 1 results in approximately 200 streams. This is so the results are the same whether the mesh is defined in cm, mm, or any other scale.	`1.0`
`lc`	line color	Line color. (See APPENDIX C: Color Names.) A color index may be used in place of a color name.	`white`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`

Option	Value	Value Meaning	Default Value
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1`
`maxsteps`	positive integer	Number of integration steps taken per stream.	`50`
`method`		Integration method for computing streams.	`rk2`
	`euler`	1st order Euler integration	
	`rk2`	2nd order Runge Kutta	
	`rk4`	4th order Runge Kutta	
	`eulera`	adaptive step 1st order	
	`rk2a`	adaptive step 2nd order	
	`rk4a`	adaptive step 4th order	
`pop`	See `pop` option on page 124.		`off`
`step`	real number `> 0.0`	Integration step size. This parameter is scaled according to the size of the mesh compared to the magnitudes of the vectors to allow the step size to give similar quality results in a wide variety of meshes. If using an adaptive method, the specified step is used only as an initial value and will be adjusted according to the curvature of the flow at each point.	`0.1`
`var`	variable name	Name of the vector variable. The variable defines pseudocoloring as well. A 3d vector in a 2D mesh is interpreted as {x_component, y_component, colorvar}.	`velocity`

Description:

Streams work in 2D and 3D meshes. Streams (or streamlines) are curves in a vector field which are tangent to the flow (instantaneous flow). They are generated from a starting "seed" point, and are propagated through a vector field via various methods.

Streams are implemented for all mesh types except non-colinear (curvilinear) 3D. For unstructured meshes, only hexahedral zones are supported at this time.

The `stream` command is similar to the `ribbon` command, except that it generates lines (streamlines) while `ribbon` generates a ribbon (or tube), which is the aggregate of two (or more) streamlines.

Examples:

```
stream numpoints=3, method="rk2a", maxsteps=40
```

See Also:

`ribbon`

surf — Set plotting options for the surface plot.

Synopsis:

`surf` *option=value [,option=value, ...]*

Arguments:

option A plot option for the surface plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`max`	real number, `off`	This value is used for coloring the variable specified in `var`. Points at or above the maximum value will be shaded with the maximum value color. A value of "`off`" indicates the actual data maximum will be used.	`off`
`min`	real number, `off`	This value is used for coloring the variable specified in `var`. Points at or below the minimum value will be shaded with the minimum value color. A value of "`off`" indicates the actual data minimum will be used.	`off`
`pop`	See `pop` option on page 124.		`off`
`scale`	`linear, log`	Display the legend scale as a linear or log scale. This option only applies when the `legend` option is on.	`linear`

Option	Value	Value Meaning	Default Value

`surfcol`	`z`
	
	color name
	
	`off`
		Shade the surface in colors per the z-values of the surface.
		Shade the surface in the specified color and light the surface. The surface is lit from front and back light sources. (See APPENDIX C: Color Names.) A color index may be used in place of a color name.
		Do not shade the surface.
			`z`

`type`	`shaded, wire, both`	`shaded` is equivalent to: `surfcol=z, wirecol=off, wire` is equivalent to: `surfcol=off, wirecol=black, both` is equivalent to: `surfcol=z, wirecol=black`	`shaded`
`var`	variable name	Name of variable to plot.	`d`
`wirecol`	color name `off`	Plot the surface as a wireframe in the specified color. (See APPENDIX C: Color Names.) A color index may be used in place of a color name. Do not plot the surface as a wireframe.	`off`

Description:

The `surf` command sets the options used when generating a surface plot. Note that this command does not change existing plots, only future ones. The surface plot only operates on 2D data. The surface is shaded using the Gouraud method.

Examples:

```
surf var=p
surf surfcol=red
```

```
surf surfcol=magenta, wirecol=blue
```

See Also:

```
defvar, iso, plot, pc
```

triad — Add or remove the miniature 3D axis image.

Synopsis:

```
triad on | off
```

Arguments:

`on`	Display a small image of the 3D axes and their current orientation.
`off`	Do not display the miniature 3D axis image.

Description:

The `triad` command indicates whether a small image of the 3D axes and their current orientation should be added to the visualization window. A value of “on” is the default, and this places the image in the lower left of the graphics visualization window. A value of “off” turns off the image. This command only works for 3D images.

Examples:

```
triad on
```

See Also:

`banner, legend`

unalias — Remove the definition of a command alias.

Synopsis:

`unalias` *name*

Arguments:

name	An alphanumeric string which was previously assigned a value via the `alias` command. This argument must start with a character.

Description:

The `unalias` command removes any definition previously assigned to *name* with the `alias` command. See the `alias` command summary for more information.

Examples:

```
unalias foo
```

See Also:

`alias`

unhide — Unhide plots.

Synopsis:

`unhide` *plot_name [, plot_name, ...]*

Arguments:

plot_name	A plot name, like `smat` or `bnd`, followed by a number which indicates the position in the list of plots. For example, if there are two smat plots, they are referenced as `smat1` and `smat2`. Plot types are: `bnd`, `iso`, `label`, `mesh`, `pc`, `pop`, `ribbon`, `smat`, `stream`, `surf`, and `vec`.

Description:

The `unhide` command unhides the specified plots, provided they were already hidden. No action occurs when `unhide` is applied to unhidden plots. The plots can be hidden using the `hide` command.

Examples:

```
unhide pc2, mesh1
```

See Also:

`hide`

unlock — unlocks attributes of windows

`unlock` *type window_id [window_id...]*

Arguments:

type	A string identifying the type of the lock. Currently only "`view`" is supported.
window_id	The identifier of the window to unlock.

Description:

The lock command allows windows to be locked together so that when one updates, all other locked windows update with it.

Currently, only a window's view may be locked.

The unlock command reverses the action of the lock command, removing windows from the list of locked windows.

Examples:

```
lock view 2 3 8
unlock view 3 2
```

See Also:

`lock`

vec — Set plotting options for the vector plot.

Synopsis:

`vec` *option=value [,option=value, ...]*

Arguments:

option A plot option for the vector plot. See table below.

value A value associated with the matching *option*. See table below.

Option	Value	Value Meaning	Default Value
`headsize`	real number `> 0.0`	Size of the arrow head as a fraction of the arrow length.	`0.25`
`lc`	line color	Line color. (See APPENDIX C: Color Names.) A color index may be used in place of a color name.	`white`
`legend`	`on, off`	Whether or not a legend should be displayed with the plot.	`on`
`ls`	`dash, dot, dotdash, solid`	Line style. (See APPENDIX D: Line Styles.)	`solid`
`lt`	`1 <=` integer `<= 4`	Number to indicate thickness. 1 is the most thin, and 4 is the thickest.	`1.0`
`numvec`	integer `> 0`	Number of vectors to plot.The total number of nodes in the problem is divided by the number of desired vectors (this number) to get a stride. This stride is used to stride through the coordinate arrays to pick out points to put vectors on.	`400`
`pop`	See `pop` option on page 124.		`off`

Option	Value	Value Meaning	Default Value
`size`	real number `> 0.0`	Magnification scale. A multiplier for manipulating the vector length.	`1.0`
`var`	variable name	Name of variable to get vector data from.	`velocity`

Description:

The `vec` command sets the options used when generating a vector plot. Note that this command does not change existing vector plots, only future ones. The variable read by the vector plot is a special merged multi-value variable which may be created with the `defvar` command.

Examples:

```
vec lc=green, size=2.5
vec var=myvelocity
```

See Also:

`plot, defvar`

vp — Set the 2D plotting viewport.

Synopsis:

`vp` *xmin, xmax, ymin, ymax*
`vp full`

Arguments:

xmin	The minimum value in the X direction. A real number.
xmax	The maximum value in the X direction. A real number. 0 <= xmin < xmax <= 1
ymin	The minimum value in the Y direction. A real number.
ymax	The maximum value in the Y direction. A real number. 0 <= ymin < ymax <= 1
`full`	Set the viewport to be as large as possible and still leave room for tick marks and their labels.

Description:

The `vp` command sets the 2D viewport. The `reset vp` command can be used to reset the viewport to its default values (0.25, 0.95, 0.15, 0.90). `full` might be used when creating output for presentations or publications.

`vp` differs from `wp`, though the two are similar. `vp` sets the portion of the window into which to put the plot. Thus its coordinates are in window coordinates. wp determines which part of the problem (mesh) to plot, and so its coordinates are in problem (“physical” or “world”) coordinates.

Examples:

```
vp .25, .75, .15, .65
vp full
```

See Also:

`reset, wp`

warning — Print a warning message.

Synopsis:

`warning` *“message”*

Arguments:

message	A quoted string which contains the warning message to print.

Description:

This command prints a warning message to the standard error, which is usually the shell window from which the program is called. This is particularly useful when you are programming your own aliases, and you want to provide a warning message. If the command is being processed in an init file, the quotes must be preceeded by a backslash.

Examples:

```
warning ”The vector plot is already plotted.”
```

See Also:

`alias, defined, echo, error, if, progn`

windeicon — Deiconify one or more windows.

Synopsis:

```
windeicon window_list
windeicon all
```

Arguments:

window_list	The list of windows to deiconify, where the window is referenced by a window id. A window id is an integer number ranging from 1 to the number of active windows. The maximum number of window is 16.
`all`	Deiconify all MeshTV windows.

Description:

Use this command to “open up” or deiconify MeshTV windows which have been iconified via the `winicon` command, or by some mechanism provided by the user’s window manager.

Examples:

```
windeicon 1 3 8
windeicon all
```

See Also:

`windel, winicon, winnum`

windel — Delete the active or a specified visualization window.

Synopsis:

```
windel
windel window_id
```

Arguments:

window_id	The window identifer, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.

Description:

If there is no argument, this command deletes the active window. If a window id is given as an argument, This command deletes the window identified by *window_id*. In either case, there must be at least one visualization window remaining after the deletion. Otherwise, the window is not deleted.

Note that windows remaining after a deletion are not renumbered. So if you had 5 windows open, and you deleted windows 3 and 4, you would be left with windows 1, 2, and 5.

If you delete the active window, the lowest numbered window becomes the new active window.

Examples:

```
windel
windel 3
```

See Also:

```
winnew, winnum, winset
```

wingrow — Make the active visualization window full size.

Synopsis:

```
wingrow
```

Arguments:

No arguments.

Description:

The `wingrow` command causes the active window to be “full” size. Full size is the size of a window when only one window is opened, as with the `winnew` command. An example of use might include issuing a `winnum 4` to generate 4 windows, and then issuing a wingrow to make the first window full size again.

Examples:

```
wingrow
```

See Also:

`winnew, winnum, winshrink`

winicon — Iconify one or more windows.

Synopsis:

```
winicon window_list
winicon all
```

Arguments:

window_list	The list of windows to iconify, where the window is referenced by a window id. A window id is an integer number ranging from 1 to the number of active windows. The maximum number of window is 16.
`all`	Iconify all MeshTV windows.

Description:

Use this command to “close” or iconify MeshTV windows. The windows iconify to the desktop rather than the MeshTV Icon Box window.

Examples:

```
winicon 1 3 8
winicon all
```

See Also:

`windeicon, winnew, winnum`

winmode — Set the cursor mode of the active visualization window.

Synopsis:

`winmode` *mode*

Arguments:

mode	One of: ‘line-out’, ‘navigate’, ‘pick’, ‘slice’, ‘slicepick’ or ‘zoom’

Description:

For 2D: The `winmode` command sets the cursor mode of the current visualization window. The default window mode for 2D objects is `navigate`, but the mode can be changed to any of the above listed modes.

In `navigate` mode, the left and middle mouse buttons have different functions. The left mouse button can be used to pan (move) the image by pressing the button and dragging the cursor across the window. Release the button when the image is in the position you want. The middle mouse button can be used to zoom in on the image either by a small amount if clicked once, or by a series of small amounts by holding down the middle mouse button. If you want to zoom by large amounts, you should select the `zoom` mode. The middle mouse button in combination with the ALT or SHIFT key depressed can be used to zoom out on the image either by a small amount or a series of small amounts as described earlier.

When a window is in `pick` mode, the cursor turns into a cross-hair. When you click the mouse in a plot, information about the selected point will be written to the command line window. This information will vary depending on the kind of plot chosen. See the `pick` command for more information.

In `line-out` (reference plot) mode, the cursor is an arrow. If you drag the cursor with the left mouse button down, you will see a line between the point where you initially clicked and the point where your cursor currently resides. When you release the left mouse button, the line will draw from your beginning point to your ending point. See the `ref` command for more information.

In `zoom` mode, the mouse can be used to select a rectangular region of interest which is then enlarged to fill the viewport. (See `zoom3` command.) The middle mouse button can be used to zoom in on the image either by a small amount if clicked once, or by a series of small amounts by holding down the middle mouse button. The middle mouse button in combination with the ALT or SHIFT key depressed can be used to zoom out on the image either by a small amount or a series of small amounts as described earlier.

For 3D: The `winmode` command sets the cursor mode of the current visualization window. The default window mode for 3D objects is `navigate`, and the other options are `zoom` and `pick`. While the `zoom` and `pick` modes works the same for both 2D and 3D, the `navigate` mode differs, and the `slicepick` mode is also available.

In the 3D `navigate` mode, the left and middle mouse buttons also have different functions. The left mouse button can be used to rotate the image by pressing the button

and dragging the cursor across the window. Release the button when the image is in the position you want. You can pan (move) the image by pressing the ALT or SHIFT key and the left button and dragging the cursor across the window. Again, release the button when the image is in the position you want. As for 2D files, the middle mouse button can be used to zoom in on the image either by a small amount if clicked once, or by a series of small amounts by holding down the middle mouse button. If you want to zoom by large amounts, you should select the `zoom` mode. The middle mouse button in combination with the ALT or SHIFT key depressed can be used to zoom out on the image either by a small amount or a series of small amounts as described earlier

The 3D pick mode works the same as the 2D pick mode, but additionally, the last zone picked can be used as the seed, or starting, zone for the segment operator.

The `slicepick` mode also allows the user to select a seed zone for the segment operator. In this mode, a slice plane will appear through the visible plots. By clicking and dragging the left mouse button up or down, the slice plane will move back and forth. Using a simple click of the mouse, a zone can then be picked. Through this interaction, the user can pick on zones on the interior of a 3D mesh. Unlike the normal `pick` mode, however, only one marker (a crosshair) will be displayed, instead of a sequence of letters.

The `slice` mode allows the user to interactively position an arbitrary slice plane in a 3D dataset. When the window is put into slice mode, the original 3D dataset is replaced with a slice plane along the axis that most faces the user. Press and hold down the left mouse button while moving the mouse in order to rotate the slice plane about its origin. Press and hold down the middle mouse button while moving the mouse to translate the slice plane along an axis perpendicular to the slice plane. The slice plane origin is moved side to side or up and down by holding down the SHIFT key and the left mouse button while moving the mouse. The slice plane origin is moved to and fro by holding down the SHIFT key and the middle mouse button while moving the mouse.

Examples:

```
winmode pick
winmode zoom
winmode line-out
```

See Also:

`defvar, pick, pickpt, pickzone, ref, refl, zoom3`

winnew — Open a new visualization window.

Synopsis:

```
winnew
```

Arguments:

No arguments.

Description:

The `winnew` command opens another visualization window. This window becomes the active window.

Examples:

```
winnew
```

See Also:

`windel, winnum, winset`

winnum — Set the number of visualization windows.

Synopsis:

`winnum` *number*

Arguments:

number	One of: 1, 2, 4, 8, 9, or 16

Description:

The `winnum` command sets the number of visualization windows used by MeshTV. Depending on the window manager used, the windows are either automatically placed on the screen for you, or you have to place each window manually. Windows, if they are arranged automatically, are arranged in rows and columns.

When you have multiple visualization windows, the command "`winnum 1`" will resize the current active window to be full-screen. The other windows will iconify. The command "winnum 4" will create four windows so that they fit in the visualization window area. If there were already more than 4 windows, the first 4 will resize and the remaining will iconify. If there were less than 4, windows will be added until 4 exist.

The `winnum` command sets the size of the windows created. New windows are created at the same size as current windows. For example, "`winnum 4`" produces 4 windows in a 2X2 grid. A subsequent call to `winnew` would produce a new window which was the same size as windows in a 2X2 grid.

Here is a list of the grid sizes:

winnum 1	1X1
winnum 2	2X1
winnum 4	2X2
winnum 8	2X4
winnum 9	3X3
winnum 16	4X4

Examples:

```
winnum 4
winnum 1
```

See Also:

`windel, winnew, winset`

winset — Set the active MeshTV visualization window.

Synopsis:

`winset` *number*

Arguments:

number	The window identifer, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.

Description:

The `winset` command tells MeshTV which visualization window is active. Only one visualization window is active at any one time. Commands for setting options, databases, plot selections, and so forth apply only to the active window. Each visualization window has its own set of options which must be set separately.

Examples:

```
winset 2
```

See Also:

`windel, winnew, winnum`

winshrink — Make the active visualization window normal size.

Synopsis:

```
winshrink
```

Arguments:

No arguments.

Description:

The `winshrink` command causes the active visualization window to be shrunk to "normal" size. This command should be issued after a `wingrow` command has been issued, and it will return the window to the size it was when it was created.

Examples:

```
winshrink
```

See Also:

`wingrow`

winzoom — Zoom on the plots in a specified visualization window.

Synopsis:

`winzoom` *window_id* x_1 y_1 x_2 y_2

Arguments:

window_id	The window identifer, an integer ranging from 1 to the number of active windows. The maximum number of windows is 16.
x1, y1	The (x,y) coordinates of the upper left-hand corner of the zoom rectangle. These are real (floating point) numbers which are problem coordinates.
x2, y2	The (x,y) coordinates of the lower right-hand corner of the zoom rectangle. These are real (floating point) numbers which are problem coordinates.

Description:

The plots in the visualization window specified by `window_id` are replotted using the zoom rectangle as the limits of the plot, thereby effecting a zoom. This command works only for 2D plots.

Examples:

```
winzoom 1 -4.0 4.0 2.0 2.0
```

See Also:

`winmode, zoom3, zoomf3`

wp — Set the 2D plotting window via problem coordinates.

Synopsis:

`wp` *xmin xmax ymin ymax*

Arguments:

xmin	The minimum value to use in the X direction. A real number.
xmax	The maximum value to use in the X direction. A real number. *xmin* < *xmax*
ymin	The minimum value to use in the Y direction. A real number.
ymax	The maximum value to use in the Y direction. A real number. *ymin* < *ymax*

Description:

The `wp` command sets the 2D plotting window using problem (physical/world) coordinates. This command allows the user to view a subset of a mesh or plot. The `rect` mode of the `winmode` command performs the same function, but allows the user to use the mouse for window selection.

`wp` differs from `vp`, though the two are similar. `vp` sets the portion of the window into which to put the plot. Thus its coordinates are in window coordinates. wp determines which part of the problem (mesh) to plot, and so its coordinates are in problem ("physical" or "world") coordinates.

Examples:

```
wp 0 10 0 10
wp 0. 15. 10. 40.
```

See Also:

`vp, winmode, wp3`

wp3 — Set the 3D plotting window via problem coordinates.

Synopsis:

`wp` *xmin xmax ymin ymax zmin zmax*

Arguments:

xmin	The minimum value to use in the X direction. A real number.
xmax	The maximum value to use in the X direction. A real number. *xmin* < *xmax*
ymin	The minimum value to use in the Y direction. A real number.
ymax	The maximum value to use in the Y direction. A real number. *ymin* < *ymax*
zmin	The minimum value to use in the Z direction. A real number.
zmax	The maximum value to use in the Z direction. A real number. *zmin* < *zmax*

Description:

The `wp3` command sets the 3D plotting window using problem (physical/world) coordinates. This command allows the user to “window in” on a smaller area of a mesh or plot. This differs from zooming since it also pulls in the 3D bounding box. Note that the portions of the problems outside the bounding box are still visible..

Examples:

```
wp3 0 10 0 10 0 10
wp3 0. 15. 10. 40. 15. 15.
```

See Also:

`wp`

zoom3 — Set the 3D relative zoom factor.

Synopsis:

`zoom3` *factor*

Arguments:

factor	Any positive real number, indicating the multiplication factor applied to the object before displaying.

Description:

The `zoom3` command enlarges or reduces the current picture by the given factor, *relative to the current picture size*. That is, each `zoom3` command operates on the current view system, thus providing a cumulative effect. As an example, each successive use of the command `“zoom3 2”` would double the size of the displayed picture.

Examples:

```
zoom3 2   # Size doubles.
zoom3 2   # Doubles size again.
zoom3 .5  # Size shrinks to half the current size.
```

See Also:

`mapping, pan, panf, winmode, zoomf3`

zoomf3 — Set the 3D absolute zoom factor.

Synopsis:

`zoomf3` *factor*

Arguments:

factor	Any positive real number, indicating the multiplication factor applied to the object before displaying.

Description:

The `zoomf3` command enlarges or reduces the current picture by the given factor, *relative to the original picture size*. Because the command does not provide a cumulative effect, multiple invocations of the command `“zoomf3 2”` would not alter the size of the displayed picture after the first invocation.

Examples:

```
zoomf3 2  # Size doubles.
zoomf3 2  # Size doesn’t change.
zoomf3 .5 # Size shrinks to half original size.
```

See Also:

`mapping, pan, panf, winmode, zoom3`

! — Re-execute a MeshTV command.

Synopsis:

```
!!
!number
!string
```

Arguments:

!	Re-execute the last command.
number	An integer number indicating the sequence number of the command to re-execute.
string	A string of characters indicating the initial letters of the command to re-execute. The last command which began with the characters in *string* will be re-executed.

Description:

The ! command is modeled after the similarly named UNIX™ C-Shell command. It allows you to re-execute MeshTV commands quickly and easily. Command selection can occur in three ways. First, by indicating the sequence number of the command (which can be retrieved by using the `history` command, or by looking at the sequence number built into the MeshTV command prompt). Second, by indicating the first few characters of the command to re-execute. The final way is an abbreviation for re-executing the last command issued to MeshTV, accessed with the command "!!".

Using this command can save typing time.

Examples:

```
!!   # Re-execute the last command
!20  # Re-execute command number 20
!pl  # Re-execute the last command beginning with 'pl'
```

See Also:

`history`

— MeshTV comment character.

Synopsis:

#
text

Arguments:

text	The text for the comment.

Description:

The character '#' is MeshTV's comment character. On a given input line, any text following a '#', till the next newline, will be ignored. This feature is typically used to annotate command input files which could be used to run MeshTV in batch mode.

Examples:

```
plot bnd  # Generate a boundary plot.
```

See Also:

`source`

APPENDIX A MeshTV Tutorial

The following tutorial illustrates some of the basic concepts and uses of the command line interface to MeshTV.

A.1 Before You Can Start

Before starting on the tutorial, you will need the following:

- A SILO file, preferrably “rect2d.silo” which can typically be retrieved at `/usr/local/meshtv/data/rect2d.silo`
- Access to the executable `meshtvx`. If when you type `meshtvx` you get the message:
  `meshtvx: Command not found.`
  then `meshtvx` is either not on your machine or not in your search path. Contact your system administrator for assistance.

A.2 Typographical Conventions

For the purpose of this tutorial, the following typographical conventions will be used:

`typeface`	Items printed in typeface should be entered verbatim.
1. normal	Numbered items printed in normal font are merely annotation and should not be entered.

A.3 Tutorial

1. Begin the command line interface to MeshTV.
   ```
   meshtvx
   ```
2. Open a SILO file.
   ```
   open /usr/local/meshtv/data/rect2d.silo
   ```

3. List the contents of the file (shows the names of all variables and meshes.)
```
ls
```
4. Plot a mesh.
```
mesh var=quadmesh2d
plot mesh
```
5. Clear the output window.
```
clear
```
6. Make a contour plot of density.
```
iso var=d
plot iso
```
7. Change the contour levels.
```
iso level={.1 1. 2.}
clear; plot iso
```
8. Change the contour levels by percent.
```
iso pct={10 25 50 75 90}
clear; plot iso
```
9. Generate a pseudo-color contour plot of pressure.
```
pc var=p
clear; plot pc
```
10. Find the min and max values of pressure.
```
minmax p
```
11. Set the range for the pseudo-color plot.
```
pc pcmin=.5, pcmax=1.5
clear;plot pc
```
12. Use the pick and query facility. Use left mouse button to select points.
```
winmode pick
pick var=p
```
13. Plot a mesh in another color on top of current plot.
```
mesh lc=red
plot mesh
```
14. Change the window limits (using physical/problem dimensions).
```
wp 0 5 0 5
```
15. Reset the physical window limits.
```
reset wp
```
16. Change to four output windows.
```
winnum 4
```

17. Change the active window.

```
winset 2
```

18. Generate a material-boundary plot.

```
bnd var=mat1
plot bnd
```

19. Change the active window and generate a solid-material plot.

```
winset 3
smat var=mat1
plot smat
```

20. Generate a distance plot of density. Use the left mouse button to select the two endpoints of a line (within window 3).

```
winmode dist
dist var=d, outwin=4
```

21. End this session of MeshTV.

```
end
```

APPENDIX B Cale aliases

Cale is a hydrodynamics code that supplies interactive graphics to its users. Since many MeshTV users also use Cale, we have provided several aliases to replicate commands that Cale users would know. We have not reproduced all of Cale's commands, but have provided an interface to what we believe are the more common commands.

TABLE 1 MeshTV Cale aliases

Cale command	Description
bigwin	Reset the window to its original extents.
lev	Set the levels for a contour plot. This command takes a list of the values at which to place contour lines.
levn	Set the number of levels for contour and pseudocolor plots. the command takes a minimum value, maximum value, and the number of levels to place between the two.
plb	Plot material boundaries.
plb2	Plot material boundaries which have thinner lines that the plb boundary plot.
plboff	Remove the boundary plot.
plc	Plot a pseudocolor plot. This command uses the first appropriate variable found in the datafile, unless a variable is specified as an argument.
plcoff	Remove the pseudocolor plot.

TABLE 1 MeshTV Cale aliases

Cale command	Description
plg	Plot grid motion labels to the nodes of the mesh. The variable name in your data file must be cgrdmv for this command to work.
plgoff	Remove grid motion labels.
pli	Plot contours. This command uses the first appropriate variable found in the datafile, unless a variable is specified as an argument.
plioff	Remove the contour plot.
plm	Plot the mesh.
plmoff	Remove the mesh plot.
plr	Plot region number labels to the nodes of the mesh. The variable name in your data file must be ireg for this command to work.
plroff	Remove the region number labels.
plv	Plot vectors. This command uses the supplied variable, if one is provided as an argument, else it uses the defvar named plv_var. This command fails if plv_var is undefined and no argument is provided.
plvscl	Scale the vector size. The first argument gives the scale, and the second optional argument specifies a variable. If no variable is provided, plv_var is used. The command fails if plv_var is undefined and no variable argument is provided.
plvoff	Remove the vector plot.
refb	Plot the boundary plot reflected across the positive X axis.
refboff	Remove the reflected boundary plot.
refc	Plot the pseudocolor plot reflected across the positive X axis. This command uses the first appropriate variable found in the datafile, unless a variable is specified as an argument.
refcoff	Remove the reflected pseudocolor plot.

TABLE 1 MeshTV Cale aliases

Cale command	Description
refm	Plot the mesh reflected across the positive X axis.
refmoff	Remove the reflected mesh plot.
refs	Plot a filled material plot across the positive X axis.
refsoff	Remove the reflected filled material plot.
refv	Plot vectors reflected across the positive X axis. This command takes the same arguments as plv.
refvoff	Remove the reflected vector plot.
rshd	Plot a filled material plot.
rshdoff	Remove the filled material plot.
winp	Set the extents of the window. The arguments to this function are xmin, xman, ymin, ymax.

APPENDIX C Color Names

The following table contains all of the color names understood by MeshTV. These color names can be used for line color attributes, text color attributes, or one of the 3D lighting attributes. This list can be extended or modified through the use of the colordef command. Up to 30 colors can be used but only the first nine colors have defined names. To use colors that do not have defined names, a color index may be used or a color name can be created with the colordef command. See the command summaries for more details.

TABLE 2 MeshTV Standard Color Table

Color Numbers	Color Names
1	red
2	green
3	blue
4	cyan
5	magenta
6	yellow
7	orange
8	violet
9	grey
None	black
None	white
None	off (used by some commands to indicate automatic cycling of colors)

The last three entries, black, white, and off cannot be modified by the user.

APPENDIX D Line Styles

The following table contains all of the line styles understood by MeshTV. These line styles can be used by many plots, such as the `bnd`, `dist`, `iso`, and `mesh` plots, to name a few. See the command summaries for more details.

TABLE 3 MeshTV's Line Styles

Line Styles
dash
dot
dotdash
solid